APRENDER VISUAL BASIC DESDE CERO

Objetivos

Los fundamentos de programación nos van a servir para desarrollar programas en distintos lenguajes. Este curso aplica los conocimientos de algoritmia y análisis a un lenguaje de alto nivel como es el Visual Basic, completando la formación básica y permitiendo introducirnos en un lenguaje.

Destinatarios

Cualquier persona que tenga suficientes conocimientos de algoritmia. Especialmente dirigido a personas que hayan realizado los cursos “Bases de la programación Nivel I y II”, pero también útil para programadores sin conocimientos de Visual Basic.

Contenidos

- ✓ INTRODUCCIÓN A PROGRAMACIÓN ORIENTADA A OBJETOS Y A VISUAL BASIC
- ✓ INSTALACIÓN. ENTORNO DE PROGRAMACIÓN. LA AYUDA DE VISUAL BASIC.
- ✓ VARIABLES CON VISUAL BASIC. ESCRITURA DE CÓDIGO.
- ✓ ESTRUCTURAS DE DECISIÓN Y DE REPETICIÓN (BUCLES) CON VISUAL BASIC.
- ✓ INSTRUCCIONES LEER Y GUARDAR. INTRODUCCIÓN A ARCHIVOS.
- ✓ CONTADORES Y OTRAS HERRAMIENTAS CON VB. FINAL DE ARCHIVO (EOF).
- ✓ PROGRAMACIÓN POR MÓDULOS CON VISUAL BASIC.
- ✓ VERIFICACIÓN DE ALGORITMOS CON VISUAL BASIC.
- ✓ ERRORES. GENERAR UN ARCHIVO PUNTO EXE. EJERCICIOS.

Duración

150 horas de dedicación efectiva, incluyendo lecturas, estudio y ejercicios.

Dirección, modalidades y certificados

El curso está dirigido por Mario Rodríguez Rancel, Jefe de Proyectos de aprenderaprogramar.com. Se oferta bajo las modalidades web (gratuito), con tickets de soporte y tutorizado on-line (material + soporte). A los alumnos que sigan el curso tutorizado on-line y cumplan el programa de trabajo se les expedirá certificado acreditativo de la realización del curso.

aprenderaprogramar.com

INDICE DEL CURSO

1. CONSIDERACIONES PREVIAS

2. ¿QUÉ ES VISUAL BASIC? ¿CUÁLES SON SUS VERSIONES? ¿ES UN BUEN LENGUAJE?

3. INTRODUCCIÓN A VISUAL BASIC

4. DESCARGAR E INSTALAR VISUAL BASIC IDE. ENTORNO DE PROGRAMACIÓN. VISUAL STUDIO.

5. EMPEZAR A TRABAJAR CON VISUAL BASIC

- 5.1 Formularios. Controles Label. Ventana Propiedades. Cuadrícula. Vista de diseño.
- 5.2 La ayuda y referencia de Visual Basic. Pensar como programadores.

6. VARIABLES CON VISUAL BASIC

- 6.1 Nombres y tipos de variables (Integer, Single, Boolean, etc.). Declaración con Dim.
- 6.2 Contenido y asignación de contenido a variables. Ejercicios resueltos.
- 6.3 Variables con índice o localizador. Arrays o arreglos
  - 6.3.1 Arrays (arreglos) unidimensionales
  - 6.3.2 Arrays (arreglos) multidimensionales

7. ESCRITURA DE CÓDIGO

- 7.1 Introducción
  - 7.1.1 Editor Visual Basic. Asistencia de Visual Basic para escritura de código. Autocompletado.
  - 7.1.2 Ordenación de líneas. Numeración de líneas. Etiquetas de líneas
- 7.2 Inserción de comentarios en el código. Rem, apóstrofe.
- 7.3 Herramientas lógicas y matemáticas en Visual Basic. Math.
  - 7.3.1 Generación de números aleatorios decimales o enteros – Función Rnd, Randomize.
- 7.4 Concatenación de órdenes y ordenación simbólica del código. Dividir líneas.
- 7.5 Instrucción Pedir con Visual Basic. Los Textbox y sus propiedades (multiline, scrollbars, etc.)
  - 7.5.1 La función Val. Obtener un valor numérico a partir de texto introducido por el usuario.
- 7.6 Instrucción Mostrar con Visual Basic. Labels y otras formas.
  - 7.6.1 Labels en Visual Basic y sus propiedades. Autosize. BorderStyle. Font
  - 7.6.2 Otras formas de mostrar mensajes: MsgBox, Print.
- 7.7 Matrices o arrays de controles con Visual Basic - .NET
- 7.8 Código asociado a eventos. Botones (Command Buttons ó Buttons). Ejemplos.
- 7.9 Ejercicios resueltos con Labels, TextBox y botones (Buttons o CommandButtons)

8. ESTRUCTURAS DE DECISIÓN CON VISUAL BASIC

- 8.1 Instrucción Si ... Entonces **(If ... Then)** y
  Si ... Entonces - SiNo **(If ... Then - Else)**
- 8.2 Paradoja del Si burlado por un intervalo
- 8.3 Instrucción Según (Caso) Hacer **(Select Case)**

9. ESTRUCTURAS DE REPETICIÓN (BUCLES) CON VISUAL BASIC

9.1 Instrucción Desde ... Siguiente **(For ... Next)** y cláusula Paso **(Step)**
9.2 Instrucción Mientras ... Hacer **(Do While ... Loop)**
9.3 Instrucción Hacer ... Repetir Mientras **(Do ... Loop While y Do…Until)**

10. INSTRUCCIÓN LEER E INSTRUCCIÓN GUARDAR. INTRODUCCIÓN A ARCHIVOS

10.1 Acceso a ficheros con Visual Basic. Secuencial, aleatorio, binario
10.2 Open, Close, Freefile. Instrucciones Write# y Input# (Una forma de guardar y leer datos)
10.3 StreamWriter y StreamReader. Write, Read, WriteLine, ReadLine. Ejemplos .NET.
10.4 Ejercicios resueltos

11.LAS HERRAMIENTAS DE PROGRAMACIÓN CON VISUAL BASIC. CONTADORES, ACUMULADORES, INTERRUPTORES, SEÑALEROS. EJERCICIOS RESUELTOS.

12. FINAL DE ARCHIVO. EOF, IS NOTHING. EJEMPLOS.

13. MODIFICACIÓN DIRECTA DEL FLUJO DE PROGRAMAS

CON VISUAL BASIC

13.1 Instrucción Finalizar. **End, Me.Close, Unload, Exit Sub.** Cerrar ventanas, subs o programas.
13.2 Instrucción SalirDesde. **Exit For**
13.3 Instrucción SalirMientras. **Exit Do**
13.4 Instrucción SalirHacer **(Exit Do)** e instrucción Hacer ... Repetir **(Do ... Loop)**
13.5 Instrucción IrA. **Go To**

14. PROGRAMACIÓN POR MÓDULOS CON VISUAL BASIC

14.1 Tipos de módulos. Procedimientos con **Sub**.
Funciones con **Function**. Parámetros. Inserción de módulos en el programa
14.2 Variables globales y variables locales. Variables de clase (class). Concepto. Ejemplos.
14.3 Módulos genéricos. Parámetros de entrada. Transferencias por defecto,
Por Referencia **(ByRef)** y Por Valor **(ByVal)**
14.3.1 Transferencias por defecto, Por Referencia y Por Valor
14.4 Arrays dinámicos y arrays estáticos. Instrucción
Redimensionar **(Redim y Redim Preserve).** Instrucción **Erase.**
14.5 Manejo de datos con arrays. Límites superior (**Ubound**) e inferior (**Lbound**)
14.6 Ejercicios resueltos

15. DEPURACIÓN DE ALGORITMOS UTILIZANDO VISUAL BASIC

15.1 Construir tablas de variables a través de Visual Basic usando Labels y TextBoxes.
15.2 Prueba (test) de programas. Programación rápida. Verificación funcional y total.
Verificación ordenada y aleatoria. Recarga de formularios.
15.3 Paso a paso y puntos de interrupción (breakpoints)
15.4 Enfoque al resultado. Enfoque al proceso. Enfoques zonales

16. ERRORES

16.1 Tipos de errores: de compilación, de ejecución y de sintaxis. Depuración. NeuN.
16.2 Captura y gestión de errores. Instrucción On Error. Objerto Err. Método Clear
16.3 Resume Next. Err.Number. Err.Description.

17. GENERAR UN ARCHIVO PUNTO EXE

18. CONSIDERACIÓN FINAL EN TORNO A VISUAL BASIC

19. ANEXO: EJERCICIOS COMPLEMENTARIOS

INDICE DE EJERCICIOS RESUELTOS

Ejercicio	Aplicación
Ejercicio nº 1	Declaración de variables (Visual Basic)
Ejercicio nº 2	Localizadores. Arrays (Visual Basic)
Ejercicio nº 3	Generación de números aleatorios (Visual Basic)
Ejercicio nº 4	Generación de números aleatorios (Visual Basic)
Ejercicio nº 5	Pedir, Mostrar y Command Buttons
Ejercicio nº 6	Pedir, Mostrar y Command Buttons
Ejercicio nº 7	Pedir, Mostrar y Command Buttons
Ejercicio nº 8	If ... Then – Else
Ejercicio nº 9	If ... Then – Else
Ejercicio nº 10	Select Case
Ejercicio nº 11	For ... Next
Ejercicio nº 12	For ... Next
Ejercicio nº 13	Acceso a ficheros (Visual Basic)
Ejercicio nº 14	Acceso a ficheros (Visual Basic)
Ejercicio nº 15	Contadores (Visual Basic)
Ejercicio nº 16	Acumuladores (Visual Basic)
Ejercicio nº 17	Acumuladores (Visual Basic)
Ejercicio nº 18	Acumuladores (Visual Basic)
Ejercicio nº 19	Acumuladores (Visual Basic)
Ejercicio nº 20	Interruptores (Visual Basic)
Ejercicio nº 21	Señaleros (Visual Basic)
Ejercicio nº 22	Señaleros (Visual Basic)
Ejercicio nº 23	Entradas controladas por el usuario (Visual Basic)

CONSIDERACIONES PREVIAS

Diferentes personas que han realizado los cursos de fundamentos de programación de aprenderaprogramar.com hacían como consideración final el siguiente comentario: "Está bien, hemos aprendido fundamentos, pero queremos ponernos delante de un ordenador y aplicarlos". Y precisamente esa es la filosofía de este curso que sirve como aplicación práctica y que tiene por objetivo aplicar los conocimientos de algoritmia sobre un lenguaje. Nótese que hay una diferencia entre esta propuesta y lo que sería un estudio en profundidad de una metodología de programación (por ejemplo programación orientada a objetos) o un lenguaje como *Visual Basic*. Un curso de estudio en profundidad requeriría gran extensión y dedicación y nosotros aquí nos vamos a centrar en la didáctica de la programación, no en los detalles. Puedes utilizar este curso para aplicar conocimientos previos y conocer el lenguaje Visual Basic, pero no esperes encontrar aquí todos los detalles y posibilidades del lenguaje. No vamos a entrar en definiciones ni en consideraciones abstractas como en parte sí hacemos en otros cursos. Tampoco vamos a seguir lo que sería un guión completo de estudio: nos vamos a limitar a seguir el mismo guión que hemos aplicado en los cursos de fundamentos de programación y programación modular, pero en este caso "sobre un lenguaje".

Aprender programación requiere tiempo y esfuerzo. Para hacer ese recorrido más llevadero, te recomendamos que utilices los foros de aprenderaprogramar.com, herramienta a disposición de todos los usuarios de la web (http://www.aprenderaprogramar.com/foros), y que te servirá para consultar dudas y recabar orientación sobre cómo enfrentarte a los contenidos. Entre los miembros del portal web y otros usuarios, trataremos de ayudarte para que el estudio te sea más llevadero y seas capaz de adquirir los conocimientos necesarios y avanzar como programador.

En el curso dejaremos de lado los aspectos estéticos: el mundo de las formas de presentación, los colores, imágenes, efectos visuales, etc. es atractivo y parte más o menos importante de los programas, pero no nuestro objetivo. Nos vamos a centrar en los "esqueletos" de los programas: el código que hace que funcionen (la lógica de la programación), no el que hace que sean bonitos.

Por último cabe hablar del lenguaje a utilizar. El conjunto de lenguajes y versiones de lenguajes constituyen una torre de Babel, un conglomerado de difícil seguimiento e interpretación. Lo más habitual es decantarse por estudiar lenguajes que adquieren popularidad y un uso extendido por todo el mundo, como pueden ser Visual Basic o Java. Pero incluso después de este ejercicio hay que enfrentarse a la evolución de estos lenguajes y a las variantes dentro de una versión. Por ejemplo, si usamos Visual Basic 2005 nos encontraremos con tres versiones: la estándar, la profesional y la empresarial. Ahí no acaba todo, al poco de estar usando esta versión comprobaremos que ha salido Visual Basic 2008, que tenderá a dejar obsoleta a la anterior, y al poco comprobaremos que ha salido Visual Basic 2010 y al poco... En definitiva, todo es un poco caótico. Nuestra opinión consiste en dejar esa carrera loca, quizás necesaria, para los programadores profesionales y empresas, mientras que los usuarios podemos contentarnos con programas que funcionen, aunque sean un poco "antiguos".

VERSIONES DE VISUAL BASIC

Visual Basic es un lenguaje desarrollado por Microsoft a principios de la década de los noventa y que se ha seguido usando, en distintas versiones, hasta hoy día. Tras su lanzamiento adquirió gran popularidad

©2006-2029 aprenderaprogramar.com

porque Microsoft facilitó un entorno de programación amigable, que era de fácil uso para los programadores y permitía crear aplicaciones para Windows con aspecto profesional y con rapidez. A lo largo de la historia ha tenido distintas versiones:

Visual Basic 1.0, 2.0 y 3.0: podemos denominar a estar versiones como antecedentes del Visual Basic que conocemos hoy en día.

Visual Basic 4.0 y 5.0: fueron versiones que incrementaron la popularidad de Visual Basic y se integraron en los sistemas operativos de última generación de Microsoft.

Visual Basic 6.0: fue la consagración de Visual Basic y supuso su expansión a gran escala. Visual Basic 6 se usó para el desarrollo de numerosas aplicaciones profesionales y empresariales por todo el mundo y tuvo un gran éxito. Tanto, que aunque ya no cuenta con soporte por parte de Microsoft, es un lenguaje que sigue siendo usado debido a que numerosos programadores estaban muy especializados en él y a que muchas empresas tenían programas funcionando con este lenguaje y no han querido eliminar estos programas.

A partir de 2002 Visual Basic se integró dentro de una plataforma de programación más amplia que admitía la programación en distintos lenguajes denominada Visual Studio.NET y empezó a usarse el término Visual Basic.NET para hacer referencia a estas nuevas versiones. No obstante, Visual Basic ha seguido siendo un lenguaje propio que ha seguido usándose. Ya dentro de Visual Studio pueden destacarse estas versiones:

Visual Basic 2005: introdujo cambios importantes en relación a la filosofía de programación en que se basaba Visual Basic (aunque el lenguaje continúa siendo esencialmente el mismo)

Visual Basic 2008, 2010, 2012, 2020… : Visual Basic es un lenguaje vivo que se mantiene integrado dentro de la plataforma Visual Studio y del que continúan apareciendo versiones que van introduciendo progresivas mejoras.

¿QUÉ VERSIÓN DE VISUAL BASIC USAR PARA SEGUIR ESTE CURSO?

Este curso se centra más en comprender cómo se aplican los fundamentos de la programación y la lógica de la programación que en los detalles del lenguaje. Las versiones que se pueden usar para seguir el curso son:

Visual Basic 6: puedes usar esta versión para seguir el curso, aunque en sistemas operativos Windows 7 y superiores será necesario hacer algunas adaptaciones para poder instalarlo, cosa que no será necesaria si usas otra versión superior.

Visual Basic 2005, 2008, 2010, 2012, 2020…: puedes seguir el curso usando cualquiera de estas versiones. Si no sabes cuál elegir, te recomendamos usar Visual Basic 2010.

Ten en cuenta que entre las distintas versiones pueden existir pequeñas diferencias y aunque intentaremos reflejarlas no hemos podido contemplar todas ellas en el desarrollo del curso. Para poder

©2006-2029 aprenderaprogramar.com

seguir el curso te bastará con recurrir a la ayuda del programa o a los foros aprenderaprogramar.com y hacer las pequeñas adaptaciones que sean necesarias.

En resumen, los lenguajes evolucionan de forma constante, pero detrás de los aparentes cambios casi siempre se encuentra el mismo "fondo". Nosotros nos vamos a centrar en estudiar ese fondo para adquirir la lógica de la programación.

Si estás dudando si Visual Basic es un buen lenguaje para aprender a programar, nuestra respuesta es que sí, te animamos a que profundices en Visual Basic, un entorno de programación que seguro te dará grandes satisfacciones.

PROGRAMACIÓN ORIENTADA A OBJETOS Y DIRIGIDA POR EVENTOS

La programación orientada a objetos (POO) es una filosofía y metodología de programación que se popularizó en la década de los 90 desplazando a la programación estructurada tradicional, no para eliminarla o dejarla obsoleta, sino para construir sobre ella. Nosotros nos vamos a centrar en aspectos fundamentales de la programación sin entrar a estudiar la programación orientada a objetos.

Todo lo que estudiemos será de utilidad para una persona que quiera profundizar en programación orientada a objetos, aunque obviamente tendrá que añadir nuevos conocimientos.

No todas las versiones de Visual Basic usan programación orientada a objetos, pero para este curso no debes preocuparte por ello. En cambio, sí que todas las versiones de Visual Basic hacen uso de un entorno gráfico de objetos y en este sentido sí debemos tener unas pequeñas nociones sobre qué es un objeto. Un objeto es la combinación de código y datos. Un módulo (paquete de código) se ejecutará en base a que está asociado a un objeto y no de forma libre. A efectos prácticos, veremos los objetos como elementos prefabricados a los que dotamos de sentido. Supongamos que somos fabricantes de naves prefabricadas. Montamos la nave (objeto), la dotamos de personal y maquinaria con una, dos o tres líneas de proceso (código) y damos instrucciones precisas sobre qué se debe hacer cuando se transmitan determinados mensajes a la nave (p. ej. "fabricar pan"), en función del contenido o existencias de la nave (datos).

La forma de pensar ha variado respecto a la programación tradicional. En primer lugar, ahora no tenemos que construir la nave, puesto que disponemos de ella prefabricada. No vamos a generar instrucciones sobre lo que debe pasar, sino sobre lo que hay que hacer en función de los mensajes que se reciban. Por último, los datos no van a viajar libres, sino que van a estar vinculados a objetos portadores de datos.

En todas las versiones de Visual Basic también resulta de gran importancia el concepto de evento. Un evento es "algo que sucede" como que el usuario haga click con el ratón sobre un botón, o que pulse una tecla, o que transcurra un determinado periodo de tiempo, etc. Debido a la importancia que tienen los eventos en la programación con Visual Basic suele decirse que se trata de "programación guiada por eventos".

Antes que adentrarnos en un desarrollo teórico vamos a preferir empezar a hablar de Visual Basic.

©2006-2029 aprenderaprogramar.com

INTRODUCCIÓN A VISUAL BASIC

Visual Basic es un lenguaje de programación creado por Microsoft Corp. que goza de gran popularidad. Las razones para ello son su potencia o capacidades, facilidad de uso y el permitir crear programas de aspecto y funcionamiento relacionados con el también muy popular "entorno Windows". Aparte de permitirnos crear programas de aspecto profesional, nos facilitará el trabajo a la hora de utilizar o vincular aplicaciones casi tan extendidas como Windows como son Microsoft Word, Excel o Access y PowerPoint.

El término "Visual" hace referencia a que nos movemos en un entorno gráfico e intuitivo. "Basic" alude al conocido lenguaje BASIC (Beginners All – Purpose Simbolic Instruction Code) del que se conservan diversas palabras claves e instrucciones. Aparte de esto, poco tiene que ver el Visual Basic con el antiguo lenguaje de programación ya que casi todo ha cambiado, y muy especialmente la forma de estructurar los programas.

Visual Basic es un lenguaje que se apoya en objetos. A nivel teórico se podría distinguir entre programación orientada a objetos, programación basada en objetos o programación que utiliza objetos. No vamos a entrar en este tipo de disquisiciones. Iremos viendo cómo programar en Visual Basic poco a poco y con ejemplos prácticos. A nivel práctico lo que nos interesa es disponer de objetos que podemos insertar en nuestros programas sin tener que teclear una sola línea de código y que esos objetos nos van a hacer fácil y cómoda la programación.

El usuario de *Visual Basic* se enfrenta a una larga lista de términos como pueden ser: propiedad, formulario, control, evento, método, función, argumento, operador, instrucción, procedimiento, etc. que inicialmente pueden resultar un tanto abrumadores por su variedad y carácter abstracto. Trataremos de evitar el uso de "terminología abstracta" en la medida de lo posible, pero necesariamente habremos de enfrentarnos a ella y utilizarla. Una forma sencilla de hacerlo es "aprender jugando". Arrancaremos el ordenador y vamos descubriendo *Visual Basic* a través de pequeños programas de prueba.

Próxima entrega: CU00304A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

©2006-2029 aprenderaprogramar.com

INSTALACIÓN. ENTORNO DE PROGRAMACIÓN

Si no tienes instalado *Visual Basic* en tu ordenador, es necesario instalar el entorno de programación para poder comenzar a programar. La versión que debes instalar (o tener previamente instalada) en tu ordenador para seguir este curso es alguna de las siguientes:

a) Visual Basic 6
b) Visual Basic 2005
c) Visual Basic 2008
d) Visual Basic 2010
e) Visual Basic 2012
f) Visual Basic 2020…

Si no sabes qué versión usar recomendamos usar Visual Basic 2010 para sistemas operativos Windows 7 o más modernos, o Visual Basic 6 para sistemas operativos más antiguos.

Para instalar Visual Basic hemos de descargarlo desde internet (o bien disponer de un cd de instalación).

INSTALAR VISUAL BASIC 6 SOBRE WINDOWS ANTERIOR A WINDOWS 7

Para instalar Visual Basic 6 coge el cd de instalación (o descarga el programa desde https://docs.google.com/file/d/0BzAA6hbbWbAeOUJkY2xnMlptRHc/edit?usp=sharing).

Pulsa sobre el archivo setup.exe con el botón derecho del ratón, elige “Ejecutar como administrador” y sigue las instrucciones hasta que se complete la instalación.

INSTALAR VISUAL BASIC 6 SOBRE WINDOWS 7, WINDOWS 8 O POSTERIOR

Para instalar Visual Basic 6 sobre Windows 7 de 64 bits o posterior, sigue las instrucciones que se indican aquí: http://www.aprenderaprogramar.com/foros/index.php?topic=286.0.

INSTALAR VISUAL BASIC 2010

Para instalar Visual Basic 2010 te hace falta el cd de instalación o el programa de instalación que se puede descargar desde internet. Si no tienes el cd de instalación puedes realizar la descarga desde:

a) http://www.microsoft.com/visualstudio/esn/downloads#d-2010-express (Escoger Visual Basic 2010 Express)

b) Alternativamente: http://visual-basic-2010.en.softonic.com/

©2006-2029 aprenderaprogramar.com

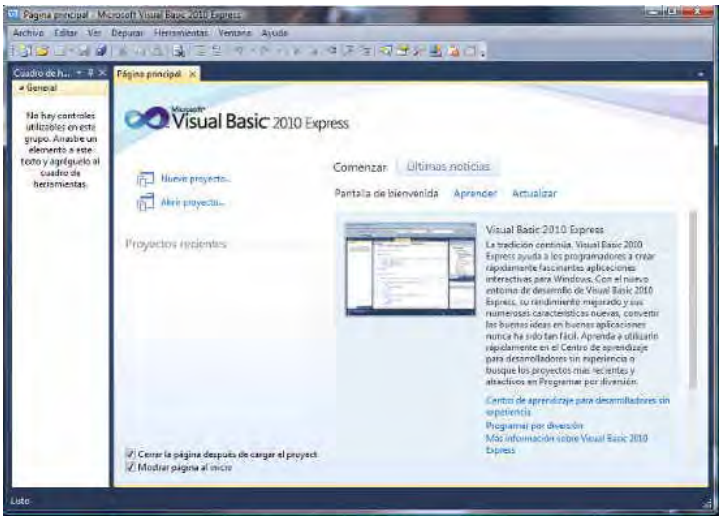
INSTALAR OTRAS VERSIONES DE VISUAL BASIC

Para instalar otras versiones: usa el cd de instalación de la versión que quieras instalar o descarga la versión express (gratuita) desde http://www.microsoft.com/visualstudio . Es posible que se te ofrezcan distintas versiones de Visual Studio (para web, para Windows Phone, Foundation Server, Desktop, etc.). Debes elegir la versión "Desktop", para desarrollo de aplicaciones de escritorio. Ten en cuenta que si instalas Visual Studio estás instalando un entorno de programación que permite programar en otros lenguajes además de Visual Basic, aunque nosotros vamos a hablar exclusivamente de Visual Basic.

Para seguir este curso no debes usar ninguna versión anterior a Visual Basic 6. Por ejemplo no son válidas las versiones Visual Basic 5 ni Visual Basic 4.

ARRANCAR EL PROGRAMA

Una vez instalado, arrancaremos el programa de la misma manera que lo hacemos con cualquier otro. La pantalla inicial del programa puede variar según la versión que hayamos instalado.



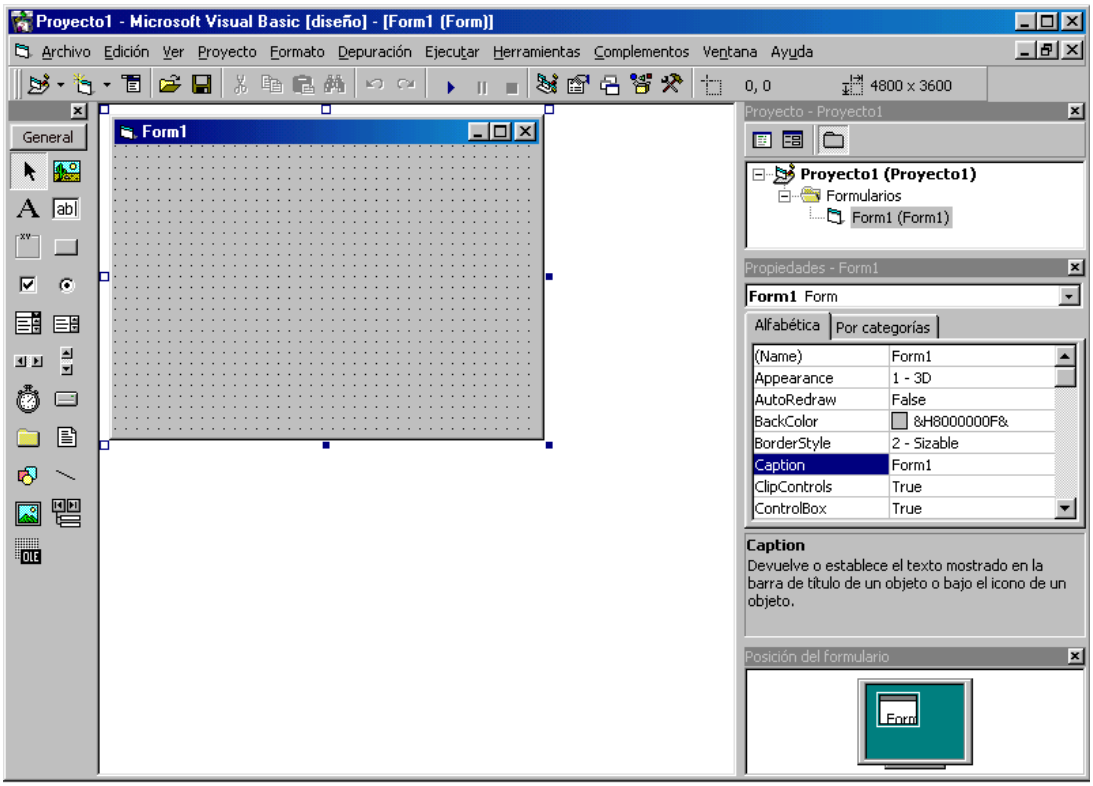
©2006-2029 aprenderaprogramar.com

En el menú Archivo seleccionamos la opción “Nuevo proyecto”. A continuación elegimos:

a) Para las versiones que nos dan la opción “Aplicación de Windows Forms” elegimos esta opción.
b) Para las versiones que nos dan la opción “Exe estándar” elegimos esta opción.

> [1]Visual Basic tiene muchas posibilidades para el desarrollo de aplicaciones, incluso aplicaciones Web. Nosotros vamos a limitarnos a usar aplicaciones de escritorio (desktop).

Una vez hemos seleccionado el tipo de proyecto entramos en el programa en sí o lo que se denomina entorno de desarrollo, es decir, un programa en el que a través de menús e iconos vamos a ser capaces de generar código en Visual Basic. El aspecto puede variar según la versión:



El aspecto es similar al de cualquier programa en entorno Windows, por lo que ya cuentas con conocimientos que te facilitan el poder moverte dentro de Visual Basic. Analizando la pantalla inicial podemos ver una barra de menús donde muchos de sus componentes nos son conocidos por ser comunes a muchos programas: Archivo, Edición, Ver, etc. En cambio otros componentes puede que no nos resulten tan comunes como Proyecto, Depuración, Datos, Complementos... Bajo la barra de menús encontramos una barra de herramientas con algunos iconos conocidos como el correspondiente a

[1] Dado que no vamos a desarrollar parte de los contenidos de *Visual Basic*, introducimos a partir de aquí una serie de cuadros con el símbolo i donde se indican algunos conceptos relevantes a modo informativo para que el lector interesado profundice en estos temas por su cuenta.

©2006-2029 aprenderaprogramar.com

Abrir, Guardar, Cortar, Copiar, Pegar y otros no habituales. Si nos posicionamos sobre ellos con el ratón nos aparecerá una etiqueta con su función: Agregar formulario, Editor de menús, Iniciar (F5), Explorador de proyectos...

En el lado izquierdo de la pantalla tenemos una caja o cuadro de herramientas con distintos iconos. Estos iconos corresponden a objetos prefabricados que vamos a usar cuando estemos programando. Si nos posicionamos sobre ellos con el ratón nos aparecerá una etiqueta descriptiva: Puntero, Picturebox, Textbox, Label, Frame, CommandButton, etc. Fíjate en que existe un aspa que nos permite cerrar este cuadro, cosa que podemos hacer cuando queramos ganar espacio en la pantalla. Cierra el cuadro con este aspa. A continuación, regenéralo pulsando el icono que muestra un martillo y una llave inglesa de la barra de herramientas. Vuélvelo a cerrar y regenéralo a través de la opción Cuadro de herramientas del menú Ver (en algunas versiones se encuentra en Ver - > Otras ventanas - > Cuadro de herramientas.

En el lado derecho superior de la pantalla tenemos un pequeño árbol que recuerda al explorador de Windows y que en este caso se llama “Explorador de Proyectos” ó “Explorador de soluciones”, según la versión que estemos usando. La ventana lleva el título Proyecto-Proyecto1, WindowsApplication1 ó similar. Aunque tendremos que hablar de cómo se estructuran los programas con Visual Basic y la terminología específica, quédate con la idea por el momento de que el Explorador te muestra la organización del programa. Por ejemplo, si tienes el programa subdividido en muchos módulos. Fíjate en el aspa que nos permite cerrar el explorador. Igual que hicimos con el Cuadro de Herramientas, cierra el explorador y regenéralo primero a través del icono de la barra de herramientas y luego a través de la opción Explorador de Proyectos (u otras Ventanas - > Explorador de soluciones) del menú Ver.

En el centro de la pantalla encontramos una ventana de las habituales en Windows, con el título de Form1 y unos botones de minimizar, maximizar y cerrar. Esta ventana se llama Formulario. Es uno de los elementos importantes de la programación con Visual Basic. El formulario es portador de objetos y a través de él entran datos al programa y salen resultados del mismo. Los formularios son el fondo o base de nuestros programas. Si alguna vez vas a iniciar un programa y no sabes cómo, recuerda que para empezar a programar necesitas una base igual que un pintor necesita un lienzo. Sin lienzo no podrás pintar y sin formulario no podrás programar. El formulario puede tener una cuadrícula interior de pequeños puntos negros (que podemos hacer que se muestren o que permanezcan ocultos), así como unos cuadritos exteriores en los vértices y puntos intermedios. Modifica el tamaño del formulario pulsando con el botón izquierdo del ratón sobre estos cuadritos exteriores: agrándalo, empequeñécelo, alárgalo hacia la derecha o hacia abajo. Comprueba cómo las dimensiones aparecen en la parte superior o inferior derecha y se modifican en función del tamaño que le vayas dando al formulario.

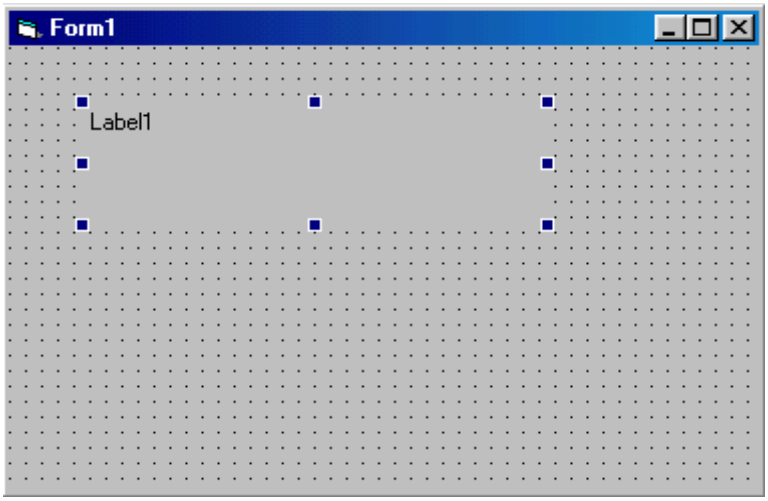
Próxima entrega: CU00305A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

©2006-2029 aprenderaprogramar.com

FORMULARIOS Y CONTROLES EN VISUAL BASIC

Seguimos explorando Visual Basic. Vete ahora al cuadro de herramientas (si no lo tienes visible deberás hacerlo visible) y busca el icono *Label* representado por una A mayúscula. Haz doble clic en él y comprobarás como aparece un objeto en el formulario sobre el que está escrito el texto "Label1". Pincha sobre él con el botón izquierdo y arrástralo para cambiarlo de posición. Para borrar un objeto pinchamos sobre él y pulsamos la tecla ***Supr*** (Suprimir). Borra el *label* y vuelve a situarte con el ratón sobre el icono de *Label*. Pulsa con el botón izquierdo del ratón una sola vez y a continuación sitúate sobre el formulario. Verás que el puntero del ratón se ha transformado en una cruz. Pulsa con el botón izquierdo del ratón sobre un punto cualquiera del formulario, y sin soltar el botón, arrastra hasta generar un rectángulo como este:



Hemos generado un label igual al anterior, aunque por distinta vía. Fíjate que estamos trabajando con elementos "prefabricados". No estamos escribiendo código para generarlos, simplemente el proceso es "Visual". Visual Basic trabaja con dos tipos de objetos principales: las ventanas, como puede ser un formulario, y los controles, como puede ser un Label (etiqueta) como el que hemos colocado en nuestro formulario. Hay distintos tipos de controles, como puedes comprobar en el cuadro de herramientas.

Un Label puede ser agrandado o empequeñecido de la misma manera que lo hacíamos con el formulario. Igualmente, las dimensiones del Label aparecen en la barra de herramientas y las podemos ver cambiar a medida que las modificamos. Pero dado que tenemos dos objetos (la ventana y el label) ¿Qué dimensiones nos aparecen en la barra de herramientas? Pulsa con el botón izquierdo sobre el formulario y verás que los cuadritos controladores de tamaño ya no están en el contorno del label, sino en el contorno del formulario, y que las dimensiones que nos aparecen ahora son las correspondientes al formulario. Es decir, hemos pasado el enfoque desde el label hasta el formulario. Igualmente podemos pasarlo del formulario al label.

Cuando modificas el tamaño de un objeto sobre el formulario, el tamaño de dicho objeto se adapta a la cuadrícula (en el caso de que tengamos la cuadrícula visible esto es fácil de ver, si está invisible no). Esto nos permite alinear objetos. Vuelve a hacer doble click sobre el icono de Label y genera otro control,

©2006-2029 aprenderaprogramar.com

que ahora te aparecerá con el nombre de Label2. Trata de poner uno debajo de otro: verás que alinearlos es fácil. Esa es la utilidad de la cuadrícula, y en general te resultará ventajoso tenerla activada. Puedes manejar la cuadrícula a través de:

a) En algunas versiones: menú Herramientas, Opciones..., Ficha General, donde dispones de un apartado denominado Opciones de la cuadrícula.

b) En otras versiones el acceso a las opciones de la cuadrícula es por otra vía. Hay que acceder al menú Herramientas, Opciones, Diseñador de Windows Forms, General, Gridsize (tamaño de la cuadrícula), ShowGrid (cuadrícula visible o no), SnapToGrid (ajustar a la cuadrícula los controles automáticamente sí o no). Además para que la cuadrícula se haga visible debemos establecer Layout Mode como “Snap to Grid”. Una vez hemos puesto todas estas opciones para que se muestre la cuadrícula es posible que todavía no se muestre. En este caso debemos cerrar la pestaña del formulario (Form1.vb[Diseño]*) pulsando en el aspa de cierre y abrirla nuevamente haciendo doble click en el archivo Form1.vb del explorador de soluciones que tendremos en la parte derecha de la pantalla (si no lo tenemos deberemos hacerlo visible).

Con las opciones de la cuadrícula podemos modificar el tamaño de la cuadrícula, hacerla visible o no visible y activar o desactivar la opción de que los controles sean forzados a situarse en posiciones de la cuadrícula. Si te resulta un poco molesta la presencia del punteado, lo mejor será que desactives Mostrar Cuadrícula (ShowGrid), pero que continúes con Forzar controles a la cuadrícula (SnapToGrid) activado. Juega activando y desactivando estas opciones para que conozcas las posibilidades que te brindan.

Pasemos ahora a la ventana Propiedades que aparece en la parte central derecha de la pantalla, debajo del Explorador de Proyectos o Explorador de Soluciones (si no la ves debes activarla). Pincha alternativamente sobre el formulario, el label1 y el label2 y comprobarás que el contenido de la ventana cambia, teniendo en cabecera el texto "Form1", "Label1" ó "Label2". Igual que para el caso de las dimensiones del objeto en la barra de herramientas, el contenido de la ventana Propiedades depende de qué objeto se encuentra enfocado. El enfoque lo controlamos pinchando sobre el objeto, pulsando la tecla TAB (tabulador) ó a través del combobox desplegable que nos permite seleccionar el objeto en la misma ventana de propiedades. Pulsa sobre el formulario y observa el contenido de la ventana Propiedades. Cada línea que aparece es una propiedad del objeto.

Pulsa en esta ventana sobre ForeColor. Verás como en la parte inferior de la ventana te aparece lo siguiente:

> *ForeColor*
>
> Devuelve o establece el color de primer plano usado para mostrar textos y gráficos en un objeto (este texto depende de la versión de Visual Basic que estemos usando. En otros casos puede ser “Color de primer plano de este componente, utilizado para mostrar texto” o algo similar).

©2006-2029 aprenderaprogramar.com

El objeto Form1, que es un formulario, tiene una serie de propiedades que te aparecen en esa lista. Busca la propiedad Width (usa la barra de desplazamiento vertical para poder acceder a visualizarla; en algunas versiones se halla como Size - > Width), cuya descripción será: "Devuelve o establece el ancho de un objeto" y establécela en 2000 (pulsa a la derecha de Width, reemplaza el número existente con 2000 y pulsa enter). Comprobarás que la ventana ha modificado su tamaño. Vete ahora al formulario y sobre los cuadritos externos, modifica el ancho. Comprobarás ahora que la cifra 2000 que habías establecido en la ventana Propiedades para Width ha cambiado. Muchas veces vas a disponer de distintas vías para hacer algo, con lo cual podrás elegir la que te parezca más adecuada.

Nota: algunos objetos no dejan modificar sus dimensiones si tienen establecida la propiedad autosize con valor true. Para permitir la modificación de dimensiones esta propiedad debe estar establecida como false.

Muchas propiedades permanecen con igual nombre y comportamiento en las distintas versiones de Visual Basic. Otras cambian de nombre con la versión, cambian de nombre y comportamiento o simplemente desaparecen de una versión a otra. Por ejemplo en un formulario la propiedad Caption en algunas versiones es equivalente a la propiedad Text en otras: sirve para establecer el texto que se muestra en la barra superior del formulario.

La ventana Propiedades, al igual que hemos visto con otros casos, la puedes cerrar con el aspa correspondiente y restablecerla con el icono de la barra de herramientas, la tecla F4 ó eligiendo la opción Ventana de Propiedades del menú Ver (en algunas versiones menú Ver, Otras ventanas, Ventana propiedades) . La ventana de propiedades nos muestra una lista de propiedades del objeto que tiene el enfoque, así como una descripción de la propiedad y a veces opciones de valores para dichas propiedades. Juega un poco leyendo propiedades de un form o de un label, y cambia valores cuando entiendas lo que indica la descripción. Modifica por ejemplo el contenido de Caption ó Text para el Form1 y comprueba lo que sucede.

Como último elemento por analizar en la pantalla, en algunas versiones en la parte inferior derecha encontramos una ventana denominada Posición del formulario que se puede hacer visible o cerrar según deseemos. Nos muestra dónde va a situarse el formulario sobre el que estamos trabajando cuando ejecutemos el programa. Pincha sobre el pequeño formulario que se ve en la pantalla dibujada y muévelo. Déjalo situado en el centro de la pantalla, para que cuando ejecutemos el programa nos aparezca ahí.

Si la versión que utilizamos no dispone de la opción de menú Ver – Ventana posición del formulario, esto no es problema porque podremos hacerlo a través de ciertas propiedades de la ventana propiedades.

Propiedad StartPosition ó StartUpPosition: nos permite elegir la posición inicial de la pantalla en la que debe aparecer el formulario. Por ejemplo elige Center Screen, el formulario se posicionará en el centro de la pantalla. Si elegimos la opción manual nosotros podremos indicar dónde debe situarse el formulario usando otras propiedades.

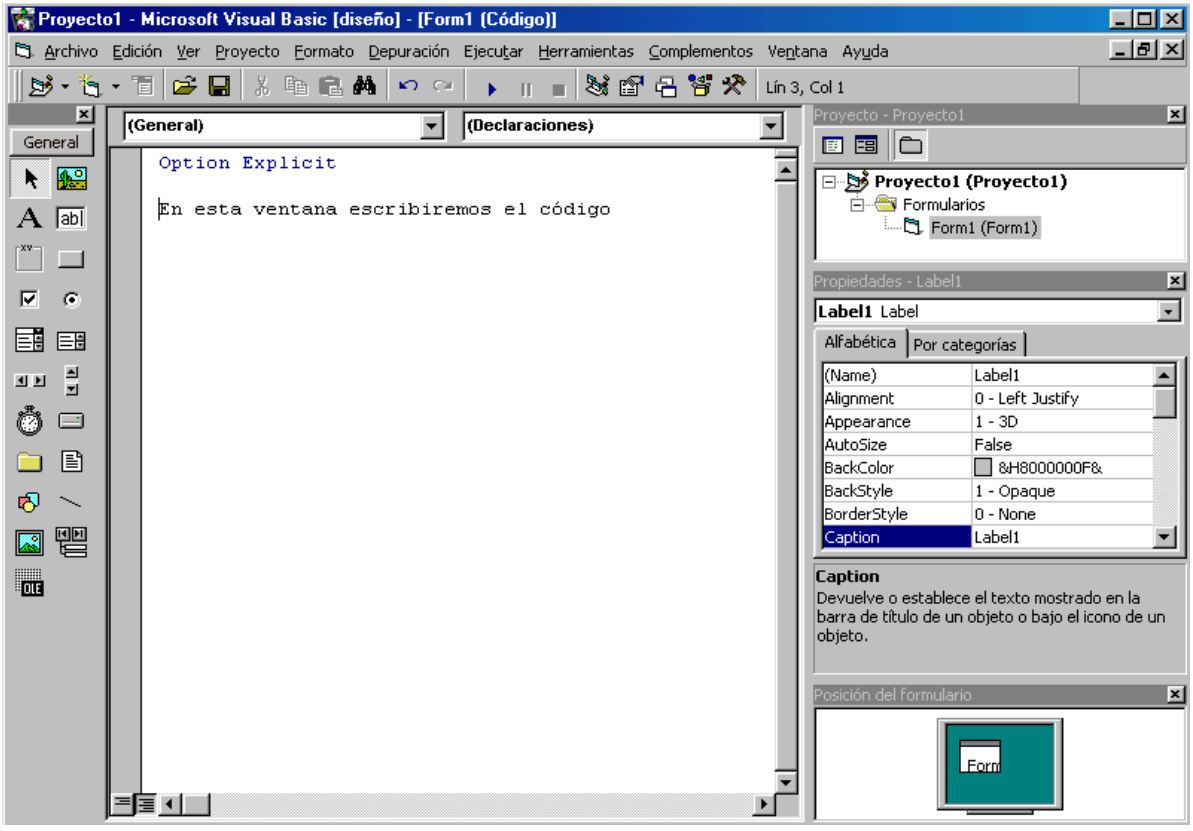
Indicar manualmente la posición del formulario se hace a través de las propiedades Top y Left en algunas versiones, o mediante las propiedades Location X y Location Y en otras versiones. A nosotros de

©2006-2029 aprenderaprogramar.com

momento nos resulta suficiente con elegir como posición de inicio Center Screen sin necesidad de especificar manualmente una posición de inicio usando coordenadas.

Vamos a ver dónde se escribe el código o instrucciones que queremos especificarle al ordenador. En el menú Ver, pulsa sobre Código. Verás que el formulario desaparece y que su lugar es ocupado por una ventana vacía o con algún texto breve (como Option Explicit ó Public Class Form1 End Class). En dicha ventana escribiremos el código. De momento, aprendamos cómo alternar entre ver los objetos que hemos diseñado o estamos diseñando y ver el código. En el menú Ver, pulsa sobre Diseñador (Objeto en algunas versiones), y comprobarás que aparece el formulario y desaparece el código. El mismo efecto se produce si pulsamos Shift y F7 simultáneamente. Y si ahora pulsamos F7 nos aparecerá el código nuevamente.

Ahora sitúate en el Explorador de Proyectos o Explorador de Soluciones e identifica los iconos existentes poniendo el ratón sobre ellos. Comprobarás que uno de ellos corresponde a Ver Código y otro a Ver Diseñador u Objeto. Pulsa alternativamente estos botones, y comprueba que es otra vía para pasar del código al diseño de objetos. Ten en cuenta que según la versión de Visual Basic que estés utilizando la apariencia de la ventana de trabajo puede variar.



Próxima entrega: CU00306

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

©2006-2029 aprenderaprogramar.com

AYUDA Y REFERENCIA DEL LENGUAJE

Si ya has trabajado con algún otro lenguaje de programación seguramente tendrás clara la importancia de la documentación de referencia de los lenguajes para los programadores. Pero si no es así, vamos a tratar de dejarlo claro. En primer lugar, cabe decir que prácticamente nadie conoce a fondo todos los términos y posibilidades de un lenguaje. Es como un idioma hablado: hay gente que lo chapurrea, gente que lo habla correctamente pero con escasez de recursos, gente que lo habla bien y con abundancia de recursos y expertos a distinto nivel. Pero nadie lo sabe "absolutamente todo". Por tanto, cuando estamos programando y no sabemos hacer algo o tenemos dudas tendremos que recurrir a ayudas externas como:

- La ayuda del lenguaje disponible en el propio entorno de programación.
- La referencia del lenguaje disponible en internet.
- Manuales del lenguaje y libros sobre el lenguaje publicados por los creadores del lenguaje o por otros autores.
- Cursos sobre programación, apuntes, etc. disponibles a través de universidades, webs, libros, etc.
- Otros programas.
- Otros programadores.

Elementos de uso habitual son la ayuda del lenguaje disponible en el propio entorno de programación y la referencia del lenguaje disponible en internet. Gracias a esto podemos acceder a encontrar información y recomendaciones que han sido generadas por los propios creadores del lenguaje. Las principales ventajas de la ayuda en el entorno y de la referencia en internet de un lenguaje son:

- **Rapidez:** si estamos programando y surge una duda o necesidad podremos resolverla a través de estas herramientas sin necesidad de movernos o desplazar la atención de lo que estamos haciendo.
- **Potencia:** las ayudas y referencias suelen ser bastante completas y útiles, por lo que podemos estar seguros de que van a resolver un alto porcentaje de los problemas que podamos tener con el lenguaje.

Como inconvenientes de las ayudas y referencias podemos citar:

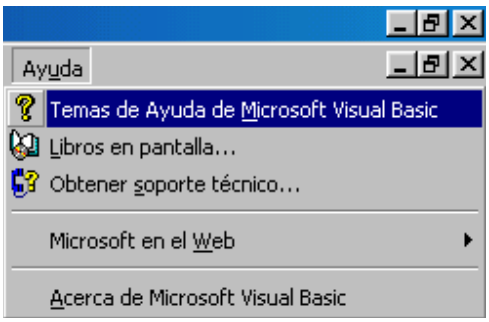
- Si no conocemos el término que describe aquello que buscamos quizás no podamos llegar a la información deseada. Podemos buscar, pero no "preguntar".
- A veces la ayuda o referencia en español es una traducción no demasiado acertada desde otro idioma, lo que dificulta su comprensión. En algunos casos es recomendable remitirse a la información en inglés.

Cualquier programador que se precie conoce y usa las ayudas y referencias de los lenguajes. Son de especial interés para programadores aprendices o autodidactas, y menos usadas por programadores avanzados o expertos, pero todos las usamos.

Veamos ahora cómo acceder y sacar partido de la ayuda y referencia de Visual Basic.

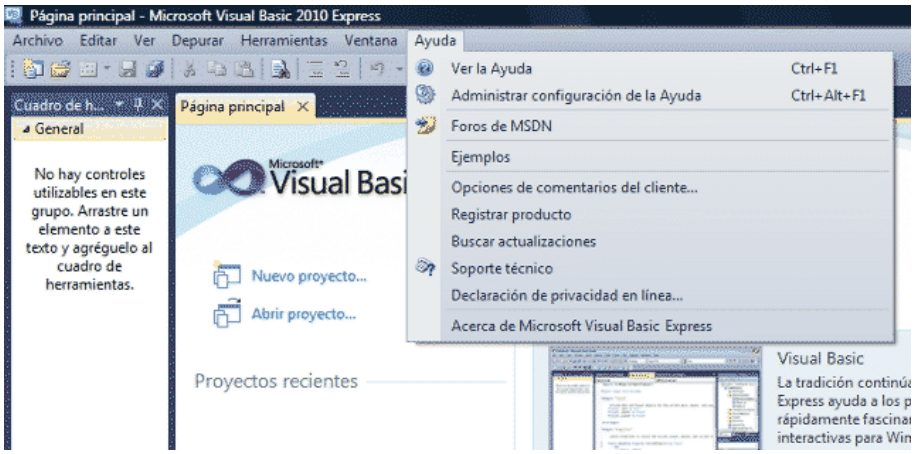
©2006-2029 aprenderaprogramar.com

En las versiones más antiguas a través de la barra de menús podremos acceder al menú Ayuda, que nos mostrará diferentes opciones.



Entrando en Temas de Ayuda podremos acceder a una completa información ordenada relativa al lenguaje. Libros en pantalla... nos dará acceso a un manual de programación con el que podremos aprender o consultar alguna parte que nos resulte de interés. La opción de soporte técnico nos informa sobre vías para conseguir parches, actualizaciones, y en general para cualquier asunto relacionado con el fabricante del software, incluido el envío de información relativa a fallos detectados en el programa o recomendaciones sobre posibles mejoras del mismo. En este caso hay un apartado específico relacionado con esto mismo a través de internet (Microsoft en el Web). Por último la opción Acerca de nos permite obtener información relativa a la versión, copyright y características del programa que estamos utilizando.

En las versiones más modernas encontraremos contenidos similares pero podemos tener otras opciones adicionales como Foros de MSDN (Microsoft Developer Network o red de desarrolladores Microsoft) desde donde podemos acceder a foros donde podemos plantear preguntas relacionadas con el lenguaje o problemas que nos surjan durante el desarrollo de programas. Además al elegir “Ver la ayuda” nos dará opción a usar la ayuda de internet o usar la ayuda local almacenada en el propio equipo. Si vamos a trabajar con frecuencia con la ayuda no es mala idea elegir Administrar configuración de la ayuda, Instalar contenido desde internet y elegir agregar Visual Basic en español. Si no se hace uso frecuente en general, usaremos la opción de internet.

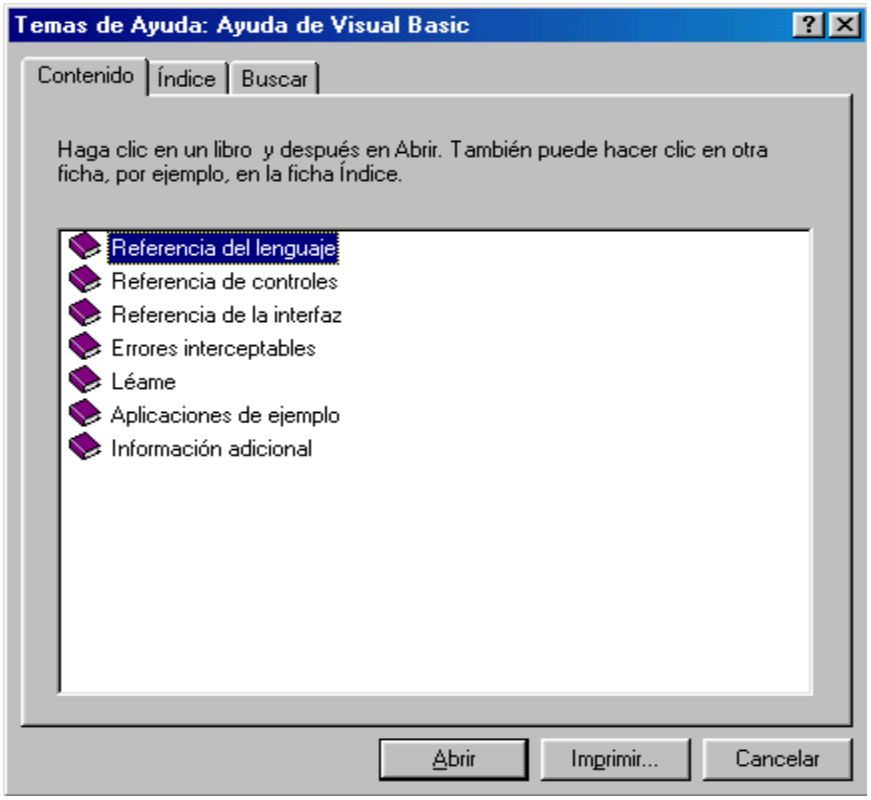


También podemos acceder a la ayuda o referencia de Visual Basic a través de la dirección web de Microsoft http://msdn.microsoft.com/es-es/

©2006-2029 aprenderaprogramar.com

BUSCAR AYUDA SOBRE UN TÉRMINO CONCRETO

Podemos acceder a información rápidamente sobre un tema determinado. En las versiones más antiguas cuando entremos a temas de ayuda nos aparecerá una ventana con tres pestañas similar a esta:

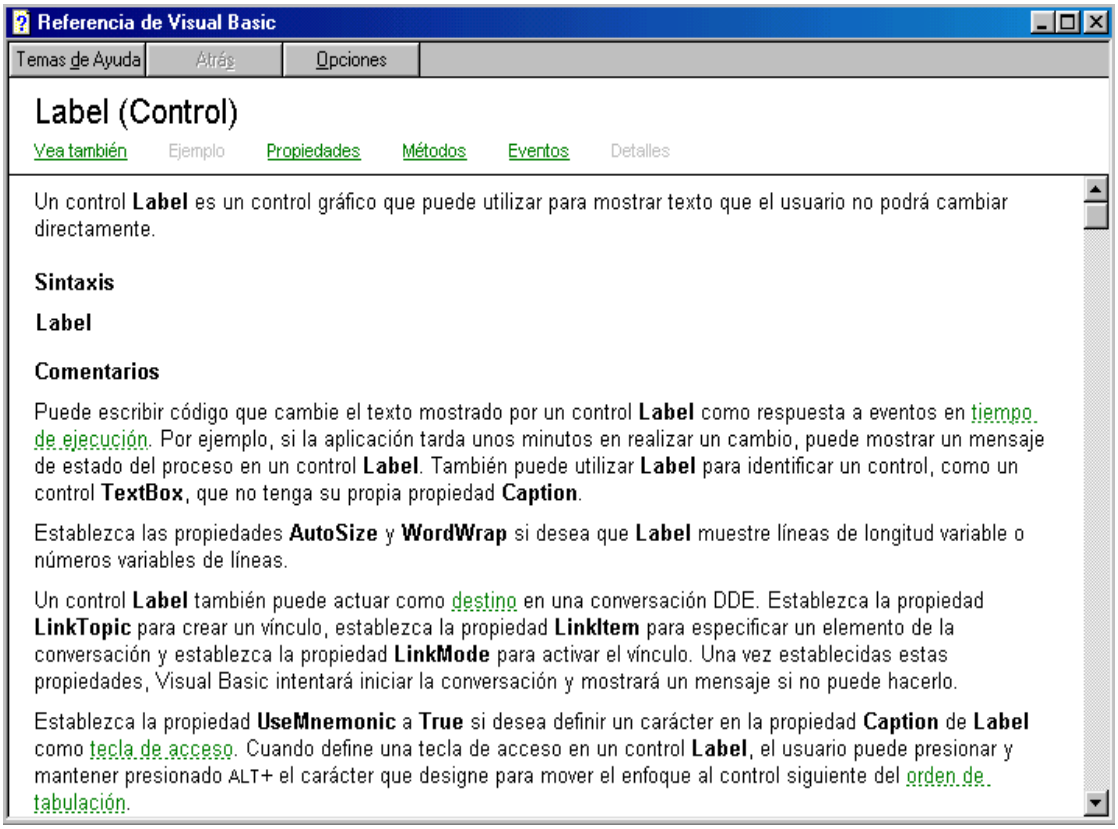


La pestaña Contenido nos permite obtener información ordenada en forma de árbol, mientras que a través de Índice podemos encontrar información ordenada alfabéticamente y a través de Buscar podemos acceder a una búsqueda de términos o frases.

En las versiones más modernas la forma más cómoda de acceder es desde el menú de Visual Studio, elegir “Ver la ayuda”, con lo que se nos abrirá una página web. Dentro de esta web elegiremos Documentación – Visual Studio, Lenguajes de Visual Studio, Visual Basic y Visual C#. Desde aquí podemos acceder a tutoriales de introducción donde se muestran los conceptos básicos, ejemplos de código, foros, etc. También podemos escribir una palabra clave como “Label” en el buscador para obtener información sobre este elemento.

Ya hemos visto que un Label es un objeto prefabricado del que disponemos directamente en Visual Basic, dentro de lo que se denominan "controles". Supongamos que queremos buscar información referida a qué es un Label y sus propiedades. En las versiones más antiguas haremos doble click sobre Referencia del lenguaje, a continuación doble click sobre Objetos, doble click sobre la letra "L" y doble click sobre Label (Control). En las versiones más modernas podemos buscar “Label” en el buscador de la página web de Microsoft. Llegaremos a una ventana con información relativa al término que hemos seleccionado.

©2006-2029 aprenderaprogramar.com



A través de la ayuda de Visual Basic tenemos acceso a centenares de ventanas de este tipo. La estructura habitual comprende:

- **Nombre del término y clasificación del mismo:** por ejemplo Label (Control) ó Asc (Función) ó Autosize (Propiedad).
- **Definición o descripción breve del cometido:** por ejemplo, "Un control Label es un control gráfico..."
- **Sintaxis:** cómo hay que escribir código relacionado con ese término. Cuando la sintaxis comprende distintos términos, se nos indica cuáles son y su descripción. En este caso, con Label no hay distintos términos por lo cual no hay descripción de partes.
- **Valores:** nos informa de valores posibles de elementos de la sintaxis y sus consecuencias. Por ejemplo, muchas propiedades se utilizan con dos valores posibles: True y False. A través de la ayuda podemos saber qué supone el que esa propiedad esté establecida en uno u otro valor.
- **Comentarios y notas:** información adicional relativa al término.

Dentro de la ventana tenemos, en la parte superior, links a distintas opciones como:

- **Vea también:** nos permite acceder a términos relacionados con el que estamos consultando.
- **Ejemplo:** nos muestra un ejemplo de uso del término seleccionado en código "real".

©2006-2029 aprenderaprogramar.com

- **Propiedades:** para un objeto, nos dice cuáles son las propiedades que tiene y nos permite acceder a información sobre ellas.
- **Métodos:** para un objeto, nos dice cuáles son los métodos que tiene y nos permite acceder a información sobre ellos.
- **Eventos:** para un objeto, nos informa de los eventos que acepta y nos permite acceder a información sobre ellos.
- **Detalles:** información adicional.

No te preocupes ahora por consultar la ayuda de Visual Basic o por llegar a ver exactamente las pantallas o descripciones que hemos mostrado a modo de ejemplo. En este curso daremos las instrucciones necesarias para que puedas avanzar sin necesidad de recurrir a ayuda suplementaria. No obstante, debes saber que un programador cuando tiene un problema recurre a la ayuda del lenguaje, su referencia en internet, libros o manuales publicados, etc. para resolverlos. Poco a poco debes ir aprendiendo a consultar y utilizar estos recursos.

Próxima entrega: CU00307A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

©2006-2029 aprenderaprogramar.com

LA AYUDA DE VISUAL BASIC

Algunos términos nos pueden parecer extraños, pero nos iremos familiarizando con ellos poco a poco. Acostúmbrate a que en torno a la información que buscas pueda haber multitud de términos y de información accesoria y que por supuesto no es objetivo de nadie saberlo todo. Se trata de aprender a obtener solución a los problemas que se nos presenten.

Navegando en la ayuda en local o en internet algunos términos aparecen en forma de links. Pinchando sobre ellos podemos obtener aclaraciones útiles. Por ejemplo, si leyendo una explicación sobre Label pinchas sobre un link que enlaza al concepto “tiempo de ejecución” aparece un mensaje aclaratorio: "El tiempo durante el cual se está ejecutando código..."

Cerremos ahora la ayuda de Visual Basic y volvamos a abrirla. En versiones más antiguas o que utilizan la ayuda local existirá una pestaña Índice. En las versiones más modernas en las que accedemos a la ayuda a través de internet podemos introducir un término en el buscador de la ayuda. Escribiendo un término en el cuadro de texto inicial o en el buscador web, nos aparecerá referenciado en la parte inferior. Escribamos Label y hagamos doble click en Label, Control (en las versiones más modernas Label (Clase) (System.Windows.Controls)). Podemos acceder a una misma información por distintas vías o distintos enlaces. Recurrir al índice o al buscador web es en general rápido y útil.

Hemos hecho un recorrido rápido del acceso a información relacionada con términos. Si accedes a la ayuda puede abrumar un poco la cantidad de información y términos extraños que aparecen (además en la ayuda web pueden aparecer distintas referencias para cada versión de Visual Basic), pero no te preocupes. Te irás familiarizando con ellos poco a poco. Juega un rato con lo que hemos visto y consulta distintos términos para familiarizarte con el manejo de la ayuda, que es nuestro objetivo en este momento. De cara al futuro, accede a la ayuda cada vez que creas que te pueda ser de utilidad. Ten en cuenta que aprender a utilizar la ayuda te permitirá ser mejor programador.

TUTORIALES O MANUALES DENTRO DE LA AYUDA

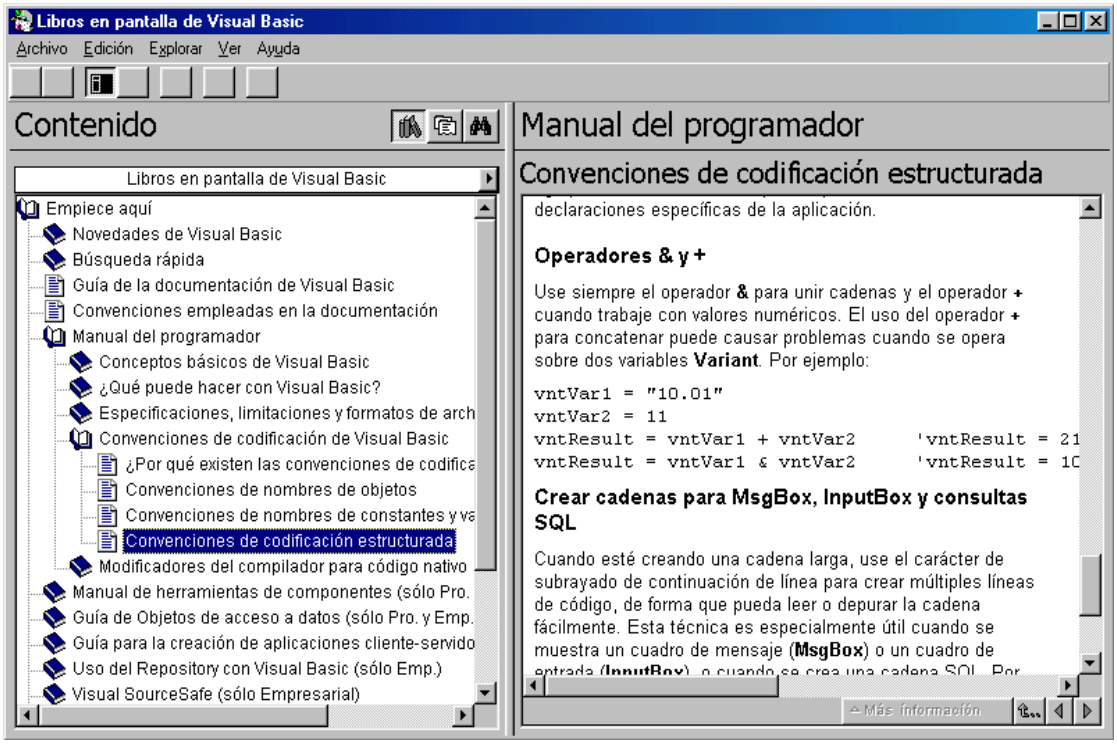
Es habitual encontrar como opción dentro de la ayuda un acceso a lo que podría ser un manual de programación, con utilidad tanto para principiantes como para programadores que llegan desde otros lenguajes de programación. En las versiones más antiguas de Visual Basic estos manuales se denominan “Libros en pantalla”, mientras que en las versiones más modernas se accede a través de un apartado de la web denominado habitualmente “Tutoriales de Visual Studio”.

En esta ayuda en forma de tutoriales podremos comenzar una lectura del manual de programación paso a paso, conociendo poco a poco el lenguaje, o bien dirigirnos a algún elemento específico que nos interese.

Supón que navegando por la ayuda seleccionas un apartado denominado "Convenciones de codificación estructurada" y que en uno de sus epígrafes encontramos recomendaciones útiles como: "Operadores & y + .Use siempre el operador & para unir cadenas y el operador + cuando trabaje con valores numéricos."

©2006-2029 aprenderaprogramar.com

Podemos aprender a programar a través de estos manuales, o sacar provecho de sus recomendaciones que van desde simples consejos relativos a cómo nombrar las variables hasta opciones avanzadas.



Además de la ayuda oficial de Visual Basic existen multitud de páginas web que también ofrecen ayuda bien sea a través de foros, tutoriales, ejemplos de código, etc., lo cual nos facilitará muchas veces el resolver problemas.

En resumen, tanto los autodidactas como los que prefieren un libro están de enhorabuena: Visual Basic es un lenguaje de amplio uso y existe mucha información sobre él. Como quien ojea un libro, recorre distintos apartados de la ayuda y lee algún trozo para familiarizarte con la estructura y el tipo de información que proporciona, que es nuestro objetivo ahora. No te preocupes ahora por memorizar nada ni por encontrar ningún término concreto, simplemente échale un vistazo a la ayuda para familiarizarte con ella. Más adelante, recurre a la ayuda cuando quieras profundizar en algún asunto que te genere interés.

Próxima entrega: CU00308A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

©2006-2029 aprenderaprogramar.com

VARIABLES CON VISUAL BASIC

Si has estudiado pseudocódigo y tipos de variables con pseudocódigo previamente los conocimientos adquiridos serán válidos y útiles con Visual Basic, pero con matices. Vamos a tratar los aspectos más básicos de la declaración de variables con Visual Basic y a dar algunas orientaciones a través de las que profundizar en el manejo de variables.

NOMBRES Y TIPOS DE VARIABLES. DECLARACIÓN. Option Explicit.

Para establecer nombres de variables puedes ver las indicaciones que se dan en el curso "Bases de la programación Nivel I" de aprenderaprogramarcom, apartado "Nombres de variables". Visual Basic permite no declarar una variable antes de ser usada. Por tanto, en cualquier punto de un programa podríamos escribir algo similar a "Mostrar velocidad01", sin necesidad de haber declarado la variable velocidad01. ¿Qué nos mostrará si no hemos asignado contenido a la variable velocidad01? Visual Basic es de los lenguajes que asignan automáticamente contenido cero, vacío o falso a una variable, dependiendo de qué tipo sea. Por tanto, se nos mostraría cero, " ", o falso. Conocido que no es estrictamente obligatorio, nosotros vamos a trabajar con Visual Basic realizando declaración de variables siempre. Los motivos para ello ya han sido expuestos, pero podemos resumirlos en:

- Generar buenos programas.
- Evitar errores y confusiones.

El propio Visual Basic dispone de una instrucción que obliga a tener que declarar todas las variables, denominada Option Explicit. La obligación de declaración de variables puede establecerse de dos maneras.

a) En la ventana de código escribir Option Explicit (en las versiones más modernas debe escribirse Option Explicit On).

b) En el menú Herramientas, selecciona Opciones... y en la pestaña Editor (en otras versiones Herramientas, Opciones, Proyectos y Soluciones, Valores predeterminados de VB) establece (o comprueba que está establecida) la opción Requerir declaración de variables como activa ó Option Explicit On.

Ten en cuenta que la opción a) es manual, y tendríamos que escribir Option Explicit cada vez que fuéramos a programar, mientras que la opción b) es automática y va a hacer que cada vez que arranquemos Visual Basic nos obligue a declarar variables sin tener que estar escribiendo código. Por tanto, activa la opción automática (o comprueba que esté activada) y así evitaremos olvidos. No vamos a volver a incidir en esto, pero siempre vamos a considerar que trabajamos con Option Explicit activado. ¿Qué ocurrirá si utilizamos en el código una variable que no ha sido declarada? En este caso Visual Basic impedirá la ejecución del programa mostrando el mensaje de error: "Error de compilación. Variable no definida / no declarada" así como la línea de programa donde ha aparecido la variable no declarada.

©2006-2029 aprenderaprogramar.com

Siempre que exista al menos una variable no declarada ocurrirá esto, lo cual nos obliga a declarar todas y cada una de las variables.

En cuanto a los tipos de variables, Visual Basic las clasifica en función del tipo de dato que contienen. Veamos las más usuales.

TIPO DE DATOS	ABREVIATURA	MEMORIA REQUERIDA	RANGO DE VALORES	EQUIVALENCIA CON USADAS EN PSEUDOCÓDIGO	OBSERVACIONES
Integer (Entero)	%	2 bytes ó 4 bytes según versión	- 32768 a 32767 ó - 2.147.483.648 a 2.147.483.647 según versión	Entero	Uso en contadores, control de bucles etc.
Long (Entero largo)	&	4 bytes ó 8 bytes según versión	- 2147483648 a 2147483647 ó - 9,2E+18 a 9,2E+18 según versión	Entero	Igual que integer pero admite un rango más amplio
Single (Decimal simple)	!	4 bytes	$-3{,}4\cdot10^{38}$ a $3{,}4\cdot10^{38}$	Real	Hasta 6 decimales. También admite enteros
Double (Decimal doble)	#	8 bytes	$-1{,}79\cdot10^{308}$ a $1{,}79\cdot10^{308}$	Real	Hasta 14 decimales. También admite enteros
Boolean (Booleano)	No tiene	2 bytes	True o False	Booleano	False es el valor predeterminado y va asociado a cero
String	$	10 bytes + 1 byte por cada carácter	0 a $2\cdot10^{9}$ caracteres	Alfanumérica	Texto de longitud variable
Variant (Object en otras versiones)	No tiene	Variable	Los del tipo double para números o los del tipo string para texto	No tiene	Admite números enteros, decimales o texto

©2006-2029 aprenderaprogramar.com

Existen otros tipos de variables. Algunos tipos son comunes a todas las versiones de Visual Basic mientras que otros tipos sólo están disponibles en algunas versiones. Otros tipos que pueden aparecer en Visual Basic son el tipo *Byte*, Char, Date, *Decimal, Object,Sbyte, Short, UInteger, Ulong, Ushort Currency, String* de longitud fija o las variables definidas por el Usuario a través de *Type ó User-Defined*. Ahora no te preocupes por conocer todos los tipos de Visual Basic, te irás familiarizando con ellos poco a poco a lo largo del curso. Puedes acceder a información sobre los tipos de variables a través de la ayuda, buscando "Tipos de datos", "Resumen de tipos de datos" y "Type (instrucción)" o "Tipo de datos definido por el usuario".

Los tipos definidos por el usuario, también llamados variables *UDT* (User Definided Type) o variables Structure (estructuras) son variables especiales que tienen un nombre "general" y luego *n* variables internas o campos. Su interés e importancia son notables. Por ello te recomendamos que más adelante profundices en ellas por tu cuenta. Por el momento solo es necesario que atiendas a los contenidos del curso.

Conociendo ya el significado de Option Explicit, los tipos de variables y cómo nombrarlas, veamos ahora cómo declararlas. Hay distintas formas y sitios para declarar variables con Visual Basic. Veremos ahora una forma básica que nos permita empezar a trabajar y, de paso, crear nuestro primer programa.

Usaremos para ello la instrucción Dim. Su sintaxis será:

```
Dim [Nombre de variable] As [Tipo de variable]
```

El lugar donde realizaremos la declaración de variables será la cabecera del código asociado al formulario. Así pues, crea un nuevo proyecto (Menú archivo --> Nuevo Proyecto --> Exe estándar ó Menú Archivo --> Nuevo proyecto --> Aplicación de Windows Forms). Vete a la ventana de código (Ver --> Código) donde quizás te aparezca una línea que ponga Option Explicit ó Public Class Form1 End Class. Escribe una nueva línea de código (debajo de Option Explicit o debajo de Public Class Form1) con este contenido:

```
Dim Numerodeplantas As Integer
```

Has declarado la variable Numerodeplantas como tipo entero. También puedes usar la abreviatura correspondiente al tipo Integer de esta forma:

```
Dim Numerodeplantas%
```

Usar el símbolo % unido al nombre de la variable declarada equivale a escribir "As Integer", lo cual nos supone ahorrar unas cuantas pulsaciones de teclas.

©2006-2029 aprenderaprogramar.com

En algunas versiones se permite declarar una variable para la que no se especifica tipo de forma extendida o abreviada. Si esto se hiciera y fuera permitido, la variable es automáticamente asignada por Visual Basic al tipo Variant. Así si escribimos:

```
Dim Numerodeplantas
```

Hay dos opciones: bien obtenemos un error de compilación (si la versión de Visual Basic que estemos usando no admite esta declaración), bien la variable Numerodeplantas es considerada por Visual Basic como tipo Variant. Todavía no está definido si es de tipo numérico (números) o alfanumérico (texto). En este caso la definición se producirá a posteriori cuando asignemos contenido a la variable. Otra forma de declarar la variable como tipo genérico (Variant u Object, según la versión) es usar la expresión:

```
Dim Numerodeplantas As Variant (ó Dim Numerodeplantas As Object)
```

Esta expresión no tiene abreviatura. Siempre será preferible especificar “As Variant” o “As Object” antes que declarar la variable sin especificación alguna. ¿Por qué? Simplemente porque al especificar As Variant ó As Object dejamos constancia de que nosotros, como programadores, hemos decidido crear esa variable como tipo Variant u Object. Si no se especifica puede dar la impresión de que se nos ha olvidado indicar el tipo de variable.

Las variables Variant u Object son un “caso especial”. Las podríamos calificar de potentes y peligrosas. Potentes porque a diferencia de otro tipo de variables, pueden contener distinto tipo de información como texto, números enteros o decimales, etc. y con pocas limitaciones. Peligrosas porque es fácil incurrir en un mal uso de ellas dando lugar a programas difíciles de leer o depurar y más ineficientes.

Supón que declaras: Dim edad As Variant ó Dim edad As Object, como variable destinada a contener la edad de una persona. Sabemos que la edad de una persona puede oscilar entre cero y 150 años (siendo groseros), y que sus valores no son decimales. Por tanto puede declararse como tipo Integer sin ningún problema. ¿Qué supone el declararla sin especificación de tipo (tipo Variant)?

a) Vamos a ocupar mayor cantidad de memoria al sistema cuando podríamos haber ocupado una menor cantidad. Es una falta de eficiencia y economía.

b) A la hora de liberar de errores al programa (depurarlo) no sabremos a ciencia cierta qué tipo de datos contiene la variable edad, ya que puede contener tanto el valor numérico 56 como el valor alfanumérico “cincuenta y seis”. Hacemos al programa más difícil de entender.

A la hora de declarar variables conviene seguir las pautas indicadas en el curso “Bases de la programación nivel I” y, resumidamente:

- Elegir siempre el tipo de variable más sencillo posible. Consideraremos que el grado de sencillez viene definido por la memoria requerida (a menor memoria requerida mayor es la sencillez). Esto redunda en un menor consumo de recursos del sistema y en una mayor rapidez de las operaciones que realiza el ordenador.

- Realizar declaraciones ordenadas y claras.

©2006-2029 aprenderaprogramar.com

- Evitar el uso del tipo Variant u Object en la medida de lo posible, es decir, sólo usar estos tipos cuando realmente sean necesarios.

Hasta ahora estamos viendo las bases que nos van a permitir construir programas. A medida que vayamos avanzando, empezaremos a construir pequeños programas para aprender practicando.

Próxima entrega: CU00309A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

©2006-2029 aprenderaprogramar.com

DECLARACIÓN DE VARIABLES EN VISUAL BASIC

A la hora de declarar variables conviene seguir las pautas indicadas en el curso "Bases de la programación nivel I" de aprenderaprogramar.com y, resumidamente:

- Elegir siempre el tipo de variable más sencillo posible. Consideraremos que el grado de sencillez viene definido por la memoria requerida (a menor memoria requerida mayor es la sencillez). Esto redunda en un menor consumo de recursos del sistema y en una mayor rapidez de las operaciones que realiza el ordenador.
- Realizar declaraciones ordenadas y claras.
- Evitar el uso de los tipos Variant u Object en la medida de lo posible.

Visual Basic permite declarar variables en múltiples líneas una por línea pero también varias variables en una línea. Veamos ejemplos:

Expresión abreviada	Equivalente
a) Dim i%, j%, k%	a´) Dim i As Integer, j As Integer, k As Integer
b) Dim i% : Dim j% : Dim k%	b´) Dim i As Integer : Dim j As Integer : Dim k As Integer
c) Dim i%, j%, k As Integer	c´) Dim i As Integer, j As Integer, k As Integer
d) Dim i% Dim j% Dim k%	d´) Dim i As Integer Dim j As Integer Dim k As Integer
e) Dim i As Integer Dim j%, k%	e´) Dim i As Integer Dim j As Integer, k As Integer

Las opciones a), b), c), d), e) dan lugar a un mismo resultado. Se declaran tres variables tipo Integer denominadas i, j, k. Las opciones a´- e´), escritas de forma extendida, también son válidas y con el mismo resultado. Nosotros nos decantamos por usar opciones como la c), donde en una misma línea aparezcan variables de un tipo de forma abreviada, excepto la última que se pone extendida de forma aclaratoria, y en las que las variables de una misma línea tienen algún tipo de relación. Por ejemplo tratarse de contadores para bucles.

Ten en cuenta que la declaración Dim i, j, k As Integer no declara tres variables de tipo entero de forma correcta. Esta sintaxis es imprecisa y por tanto no debemos utilizarla. Según la versión de Visual Basic que estemos usando esto puede generar errores o problemas. Conviene tener cuidado a la hora de realizar declaraciones de variables para evitar prácticas de este tipo.

Una expresión como Dim i!, j%, k& equivale a Dim i As Single, j As Integer, k As Long. Esta expresión es válida, aunque como hemos dicho preferimos no declarar distintos tipos de variables en una misma línea.

©2006-2029 aprenderaprogramar.com

EJERCICIO

Realizar una declaración de variables para las siguientes propuestas de programas.

a) Un programa que muestra un mensaje de bienvenida.

b) Un programa que nos pide nuestra edad y nos muestra el año en que nacimos.

c) Un programa que nos muestra el valor medio de tres magnitudes.

d) Un programa que trabaja con: tres contadores para bucles, tres valores de tensión en una viga (de magnitud no conocida), valores de longitud de la viga, canto y ancho y un valor interruptor que permite o impide que se muestren resultados.

SOLUCIÓN

a) Dim mensaje As String. También es válida la expresión abreviada Dim mensaje$.

b) Dim edad As Integer. También podemos usar la expresión abreviada Dim edad%.

Se podría definir otra variable llamada añonacimiento según el programador lo estime oportuno o no.

c) Dim magnitud1!, magnitud2!, magnitud3 As Single

   Dim media As Single

Hemos utilizado el tipo Single para cubrirnos las espaldas. No sabemos con qué tipo de valores vamos a trabajar y por ello cogemos un tipo de variables bastante amplio como es el Single. Si se considera necesario también puede usarse Double.

Hemos utilizado dos líneas pero sólo un tipo de variable ¿Por qué? El programador ordena la declaración de variables como estime más conveniente. En este caso ordenamos “valores para el cálculo” en una línea y “resultados” en otra. Resulta más fácil de leer y analizar que estando todo en una sola línea.

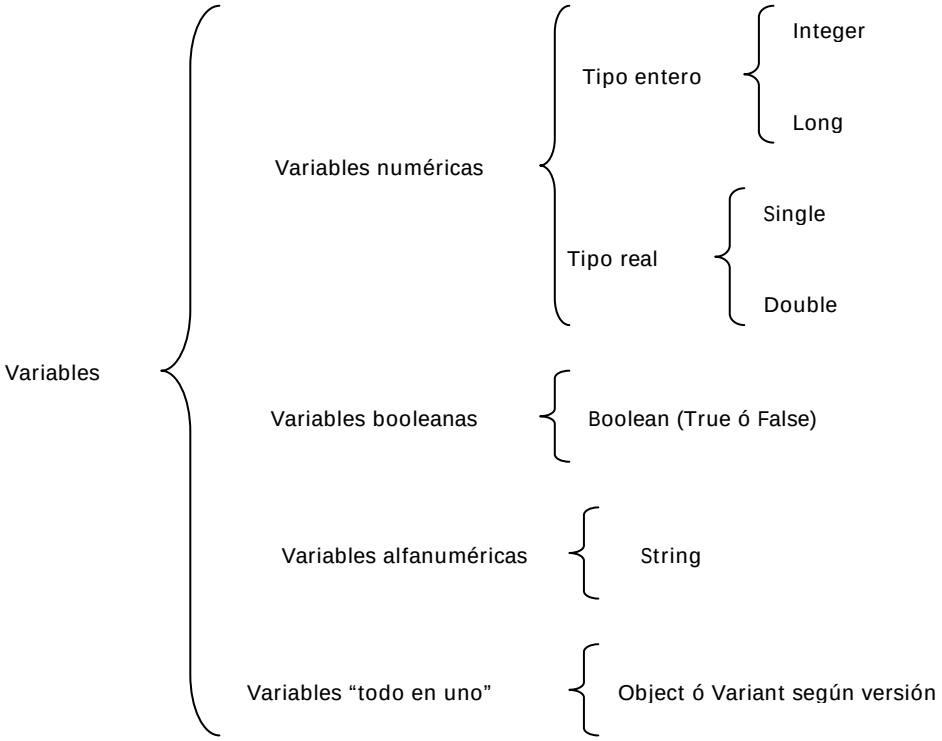
d) Dim i%, j%, k As Integer
   Dim tension1!, tension2!, tension3 As Single
   Dim largo!, canto!, ancho As Single
   Dim Muestraresultados As Boolean.

Siempre será preferible usar las expresiones tension1, tension2 y tension3 antes que t1, t2 y t3 que resultan más confusas. Sin embargo, este tipo de decisiones siempre quedan a criterio del programador.

©2006-2029 aprenderaprogramar.com

CONTENIDO Y ASIGNACIÓN DE CONTENIDO A VARIABLES

Las normas para asignar contenido a variables serán las indicadas en el curso "Bases de la programación nivel I" de aprenderaprogramar.com, teniendo en cuenta que de forma general usaremos estas equivalencias.



Ejemplos de asignación de contenidos son estos:

Declaración	Ejemplo asignación contenidos
Dim A As Integer	A = 123
Dim A As Single	A = - 3323.57
Dim A As String	A = "Hay que mejorar"
Dim A As String	A = "123 coches"
Dim A As String	A = "Son 35,37 euros"
Dim Salario As Integer	A = 30500
Dim Salario As Single	A = 30500

©2006-2029 aprenderaprogramar.com

Declaración	Ejemplo asignación contenidos
Dim Salario As String	A = “Se asignarán 30500 euros”
Dim A%, B%, Suma%	A = 5 + 2 B = 32 Suma = A + B [Suma valdrá 39]
Dim A%, B%, C%, D%, E%	A = 5 B = 32 C = A * B [C toma el valor 160] D = A + C [D toma el valor 165] E = D [E toma el valor 165]
Dim Agotamiento As Boolean	Agotamiento = True

Nota: la separación decimal normalmente es un punto, pero tendrás que comprobarlo en tu computador porque puede variar en función de las configuraciones aplicadas. Es decir, normalmente se escribirá un decimal como 3.33 en lugar de cómo 3,33.

Un programa que conste de:

```
Dim A As Integer
A = 7 * B
```

dará lugar a un error debido a que *B* no está declarada.

En cambio:

```
Dim A%, B%
A = 7 * B
```

Supone que A valga cero, ya que si no se asigna un valor a B ésta tiene por contenido el valor por defecto, es decir, cero. 7 * B equivale en este caso a 7 * 0, que vale cero.

Próxima entrega: CU00310A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

©2006-2029 aprenderaprogramar.com

CREAR UN PROGRAMA BÁSICO

Vamos a crear un programa que declare una variable tipo String llamada mensaje y nos muestre un texto en pantalla.

Para ello como primer paso abrimos un proyecto de Visual Basic. Recordar que esto se hace de la siguiente manera. En el menú Archivo seleccionamos la opción "Nuevo proyecto". A continuación elegimos:

a) Para las versiones que nos dan la opción "Aplicación de Windows Forms" elegimos esta opción.

b) Para las versiones que nos dan la opción "Exe estándar" elegimos esta opción.

Una vez abierto el proyecto debe aparecernos el formulario vacío sobre la pantalla. Hacemos doble click sobre el formulario y se nos debe abrir la ventana de código conteniendo algunas líneas. En caso de que no se te abra la ventana de código, vete al menú "Ver" y elige la opción "Código".

Escribiremos el siguiente código que corresponde al programa que vamos a ejecutar:

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim mensaje As String

Private Sub Form_Load()
mensaje = "Bienvenido a este programa"
MsgBox (mensaje)
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
   Dim mensaje As String
   Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      mensaje = "Bienvenido a este programa"
      MsgBox(mensaje)
   End Sub
End Class
```

Ten en cuenta que según la versión de Visual Basic que estés utilizando pueden existir pequeñas diferencias en el código que debe usarse. Nosotros pretendemos centrarnos en la lógica de la programación más que en los detalles de la sintaxis de una versión concreta. Si es necesario, consulta la ayuda para conocer la sintaxis exacta que debes emplear.

Analicemos las líneas que lo componen.

1. Option Explicit (Option Explicit On). Ya hemos definido su significado.
2. Dim mensaje as String, declara una variable denominada mensaje como tipo alfanumérico de longitud variable.
3. Private Sub Form Load() ó Private Sub Form1_Load (…). Esta parte de código corresponde a la apertura de un módulo de código de Visual Basic, que en la terminología del lenguaje se denomina

©2006-2029 aprenderaprogramar.com

Procedimiento. La cuestión se puede hacer un poco complicada en lo que se refiere a terminología, pero no vamos a preocuparnos por esta cuestión ahora.

En el curso “Bases de la programación I” de aprenderaprogramar.com se trabaja con programas que constan de un algoritmo principal y distintos módulos o subprogramas. La estructura y terminología de un programa en Visual Basic es algo más compleja, al trabajarse con distintos elementos dentro de un programa. Estos elementos varían según la versión, pero podrían ser por ejemplo módulos de formulario (.frm), módulos estándar (.bas), módulos de clases (.cls), etc.. Nosotros vamos a centrarnos en lo que es un módulo de formulario: un fichero en el que almacenamos información relacionada con el formulario donde hemos colocado objetos y para el que hemos escrito un código. Así pues, de momento para nosotros el programa va a constar de:

- Un formulario donde hemos colocado objetos (trabajo de diseño de tipo gráfico). Excepcionalmente un formulario puede carecer de objetos y encontrarse vacío.
- Un código que consta normalmente de: Option Explicit (Option Explicit On), Declaración de variables y Procedimientos. Un procedimiento es un conjunto de código que se ejecutará en un momento dado, algo equivalente a lo que habíamos definido con el pseudocódigo como módulo o subprograma.

Un procedimiento puede ejecutarse por varios motivos como:

a) Ser llamado desde alguna parte del programa.

b) Se produce un evento que da lugar a que se ejecute el código.

En este ejemplo que estamos viendo, el procedimiento se ejecuta cuando tiene lugar un evento que es la carga del formulario (form load), que vamos a considerar la ejecución del programa. Así pues, el código comprendido entre el fragmento de código de tipo “Private Sub Form_Load()” y “End Sub” se va a ejecutar cada vez que mandemos correr el programa, es decir, cada vez que ordenemos su ejecución.

4. mensaje = "Bienvenido a este programa", asigna a la variable mensaje el contenido de tipo alfanumérico indicado.

5. MsgBox (mensaje), indica al ordenador que proceda a mostrar el contenido de la variable mensaje dentro de una caja de mensajes.

6. End Sub, define el final del módulo de código (procedimiento).

Ejecuta el programa pulsando la tecla F5. También puedes hacerlo a través del menú <<Ejecutar / Iniciar>> ó <<Depurar / Iniciar la depuración>> según la versión que estés utilizando. En algunas versiones existe la opción <<Ejecutar / Iniciar con compilación completa>>, que permite que la ejecución del programa se haga realizando una revisión de errores más exhaustiva que si se ejecuta el programa pulsando simplemente F5.

©2006-2029 aprenderaprogramar.com

Tras ejecutar el programa te aparecerá en pantalla el mensaje Bienvenido a este programa. Para cerrar el programa pulsa aceptar y haz click sobre el aspa de cierre de la ventana. Vamos a hacer una pequeña mejora estética. Vete a Ver-Diseñador (Ver-Objeto) y cambia el valor de la propiedad Caption ó Text (según la versión que estés usando) del form a "Bienvenida" y modifica el código de la siguiente manera.

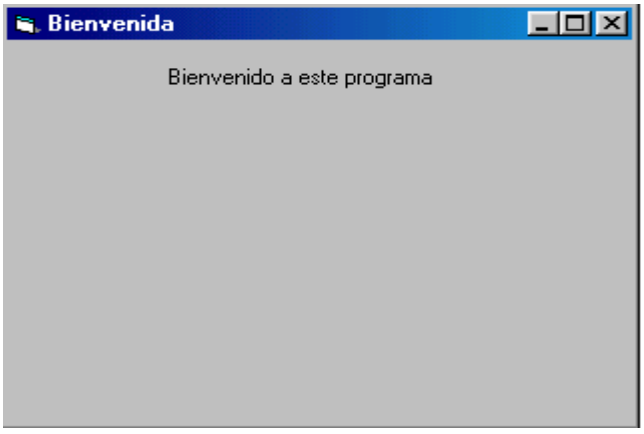
Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim mensaje As String
Dim espacios As String
Private Sub Form_Load()
      espacios = vbTab
      mensaje = espacios & "Bienvenido a
este programa" & espacios
      MsgBox (mensaje)
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
   Dim mensaje As String
   Dim espacios As String

   Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
      espacios = vbTab
      mensaje = espacios & "Bienvenido a este
programa" & espacios
      MsgBox(mensaje)
   End Sub
End Class
```

Fíjate que cuando aparezca la ventana del formulario, debido a que hemos modificado sus propiedades, ahora nos aparecerá el texto “Bienvenida” en la parte superior.



También hemos usado otros elementos que iremos explicando poco a poco a lo largo del curso. El objetivo ahora es simplemente irnos familiarizando con Visual Basic.

©2006-2029 aprenderaprogramar.com

La constante vbTab nos ha servido para introducir un tabulador. Ello da lugar a que el texto aparezca con un pequeño margen por el lado izquierdo y derecho sobre la ventanita del mensaje. Prueba a introducir variables de tipo numérico y a dar lugar a que se muestren sus contenidos sobre la ventana del mensaje. Juega con los nombres de las variables, sus contenidos y la asignación de contenidos.

También puede resultar de interés que compruebes qué ocurre si declaras una variable de un tipo y le asignas contenido de otro tipo.Por ejemplo declara *Dim salario As Integer*, asígnale el contenido *salario = "Son 3000 euros"* y prueba a mostrarlo en pantalla.

¿Qué ocurre cuando incumplimos las previsiones de *Visual Basic* para asignar contenidos a variables (por ejemplo, para una variable A tipo Integer definir A = 53200000000000000000000, que está fuera del rango previsto)? No vamos a analizar los distintos casos que se pueden presentar, sino a tratar de dar una respuesta genérica. Cuando hacemos algo no esperado, como asignar un valor fuera de rango, asignar un valor que no concuerda con el tipo de la variable, sumar variables numéricas con alfanuméricas, asignar decimales a un número entero..., etc. pueden suceder varias cosas:

a) Salta un error. Un mensaje impide que el programa comience a ejecutarse, o bien el programa se detiene mientras se está ejecutando.
b) El programa se ejecuta pero los resultados no son los deseados.
c) El programa se ejecuta y los resultados son los deseados.

En resumen, es difícil prever lo que va a suceder, aunque se pueden estudiar y manejar las circunstancias. Por ejemplo una variable que se declare como tipo Long pero a la que se asigna un contenido numérico real con dos decimales no da lugar a un error, pero sí a una pérdida de información derivada de que el número decimal se va a redondear a un entero. Las consecuencias de esta circunstancia habría que valorarlas para cada programa y circunstancias concretas.

Como programadores hemos de buscar programas 100% predecibles, en los que no se pueda producir que "sea difícil prever lo que va a suceder". Por tanto intentaremos que la declaración y asignación de contenidos a variables se ajuste a las normas de *Visual Basic*. En última instancia, podremos prever una rutina de gestión de errores para casos imprevistos.

Próxima entrega: CU00311A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

©2006-2029 aprenderaprogramar.com

VARIABLES CON ÍNDICE O LOCALIZADOR. ARRAYS.

El concepto de array con Visual Basic coincide con el que se expone en el curso "Bases de la programación nivel I" de aprenderaprogramar.com en pseudocódigo. Veremos ahora cómo declarar arrays estáticos de una dimensión. La gestión de arrays multidimensionales y dinámicos la veremos más adelante.

ARRAYS UNIDIMENSIONALES

La sintaxis a emplear será:

```
Dim [Nombre del array]([dimensión]) As [Tipo de variable]
```

Si no se declara el tipo de variable el array será tipo Variant ó tipo Object (según la versión de Visual Basic que estemos empleando) por defecto. Conviene tener cuidado con esto y seguir las mismas pautas que hemos indicado para las variables. Ejemplos de declaración de arrays serían:

- Dim Vez(10) As Integer
- Dim Amigo(1000) As String
- Dim Value(25) As Single
- Dim NúmerodeCoches(24) As Integer
- Dim Jugador(8) As Long
- Dim TCP(3) As Boolean

Cuando creamos un array del tipo A(n) podríamos pensar que estamos creando n variables que son A(1), A(2), A(3), ..., A(n). Pero en realidad estamos creando n+1 variables porque Visual Basic crea también A(0). Esto conviene tenerlo en cuenta porque puede dar lugar a alguna confusión. Disponer de un valor con índice cero puede ser de utilidad en situaciones como considerar cada variable asociada a una hora del día, empezando a contar desde la hora cero hasta la 23 (total de 24 horas), cosa que es habitual en algunos países. En lugar de 1, 2, 3, ..., 24 estaríamos usando 0, 1, 2, ..., 23.

En la mayoría de lenguajes de programación se usa el índice cero como primer índice de un array. No obstante, en las versiones menos recientes de Visual Basic se permitía establecer como primer índice de un array el 1 mediante la instrucción ***Option Base***. A través de ella se puede establecer como primer índice del array el uno en vez de el cero en aquellas versiones que lo admiten. Para ello se escribe en la cabecera del programa:

```
Option Base 1
```

Option Base afecta a todos los arrays que se declaren. También podemos indicar que el primer índice de los arrays es cero escribiendo Option Base 0 en las versiones de Visual Basic que admiten esta instrucción. Esto equivale a no poner nada (por defecto el primer índice será cero). Vamos a trabajarlo sobre el ordenador. En la ventana de código escribe lo siguiente:

©2006-2029 aprenderaprogramar.com

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Numerodecoches(24) As Integer

Private Sub Form_Load()
Numerodecoches(0) = 14
MsgBox ("El número de coches en la hora
cero fue " & Numerodecoches(0))
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
   Private Sub Form1_Load (ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Dim Numerodecoches(24) As Integer
      Numerodecoches(0) = 14
      MsgBox("El número de coches en la hora cero fue " &
Numerodecoches(0))
   End Sub
End Class
```

Al ejecutar el código (Ctrl+F5), te aparecerá el mensaje "El número de coches en la hora cero fue 14". Modifica el programa indicando Numerodecoches(29) = 14 en lugar de Numerodecoches(0) = 14. Prueba a ejecutarlo y te aparecerá un mensaje del tipo: "Error en tiempo de ejecución. El subíndice está fuera del intervalo". Efectivamente, se debe a haber puesto un índice fuera del rango y ser los valores que puedes usar en el código como localizador de una variable del array los comprendidos entre 0 y 24. Al estar usando la variable Numerodecoches(29), Visual Basic detecta la incoherencia y genera un error.

Otra opción que nos dan algunas versiones menos recientes de Visual Basic es tener un array de variables cuyos localizadores no comienzan en cero ni en uno, sino en un número específico. La sintaxis para ello en las versiones que lo permiten es:

```
Dim [Nombre de variable]([Valor inicial] To [Valor final]) As [Tipo de variable]
```

Supongamos que queremos definir una serie de variables asociadas a las horas comprendidas entre las 10 y las 18 horas. En las versiones que lo permiten podríamos crear un array del tipo:

```
Dim Magnitud (10 To 18) As Integer
```

En este caso si la versión admite esta sintaxis los valores válidos como índices del array serían 10, 11, 12, 13, 14, 15, 16, 17 y 18. Cualquier número que no sea de esta serie usado como localizador del array daría lugar a un error por subíndice fuera del intervalo. En las versiones más recientes de Visual Basic no es posible definir índices en un rango, de modo que hemos de declarar el array como un array normal cuyo primer índice es cero, pudiendo dejar vacíos o sin usar aquellos índices que no nos resulten útiles.

El contenido de una matriz estática puede ser borrado utilizando la instrucción **Erase**. La sintaxis será la siguiente:

```
Erase [Nombre de array1], [Nombre de array2], ..., [Nombre de array n]
```

©2006-2029 aprenderaprogramar.com

Ejemplos de uso de Erase pueden ser:

- Erase Númerodecoches
- Erase Númerodecoches, Usuarios, Calles

Al invocar a Erase sobre una variable de tipo array el resultado depende de la versión de Visual Basic que estemos utilizando:

a) En las versiones menos recientes el contenido de las variables que constituyen el array estático se transforma en cero si son de tipo numérico o cadena vacía si son de tipo texto (String), pero el array sigue existiendo y teniendo un contenido válido.
b) En las versiones más recientes el contenido de las variables que constituyen el array estático queda eliminado y no es posible invocar un elemento del array. La variable del array pasa a tener valor Nothing y para volver a usarla tendríamos que establecer una redefinición del array con la instrucción ReDim, que estudiaremos más adelante.

El comportamiento de Erase lo volveremos a estudiar pues tiene distintas aplicaciones. Ejecuta el siguiente programa para comprobar el funcionamiento de Erase.

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com

Option Explicit
Dim Numerodecoches(24) As Integer

Private Sub Form_Load()
Numerodecoches(0) = 14
Erase Numerodecoches
MsgBox ("El número de coches en la hora
cero fue " & Numerodecoches(0))
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
   Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Dim Numerodecoches(24) As Integer
      Numerodecoches(0) = 14
      Erase Numerodecoches
      MsgBox("El número de coches en la hora cero fue " &
Numerodecoches(0))
   End Sub
End Class
```

El resultado será:

a) Para las versiones menos recientes será: El número de coches en la hora cero fue 0. A pesar de haber definido Numerodecoches para la hora cero como 14, este valor no se llega a mostrar porque se ejecuta un Erase, que da lugar a que todos los valores del array se establezcan a cero por ser el array de tipo Integer.

b) Para las versiones más recientes: Error. No se controló NullReferenceException. Referencia a objeto no establecida como instancia de un objeto. Este error se genera porque se considera que el array no tiene índices establecidos y tendríamos que volver a establecerlos usando la instrucción ReDim que estudiaremos más adelante.

Si eliminas el Erase, te deberá aparecer en pantalla "El número de coches en la hora cero fue 14".

©2006-2029 aprenderaprogramar.com

EJERCICIO:

Crea el código de dos programas que cumplan las siguientes premisas:

a) ***Programa 1***.

Declara un array tipo Integer denominado Numerodecoches cuyo número de elementos sea 24. Declara una variable tipo Integer que se llame R. En el procedimiento de carga del formulario establece el valor de R en 2 y el valor de Numerodecoches para un localizador de valor R en 233. Procede a mostrar en pantalla un mensaje que indique cuál es la hora R y el número de coches para la hora R. Finalmente, modifica únicamente la asignación de valor a R de modo que en vez de 2 sea 21 y ejecuta de nuevo el programa.

b) ***Programa 2.***

Sobre el programa anterior realiza los siguientes cambios. Mantén el número de elementos de Numerodecoches en 24. Declara dos variables A y B de tipo Integer. Establece A con valor 8, B con valor 4 y R con valor A dividido entre B. Ejecuta el programa.

SOLUCIÓN:

El programa 1 será el siguiente. Si lo ejecutamos obtendremos "El número de coches en la hora 2 fue 233". Si cambiamos R = 2 por R=21 obtendremos "El número de coches en la hora 21 fue 233".

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Numerodecoches(23) As Integer
Dim R As Integer

Private Sub Form_Load()
R = 2
Numerodecoches(R) = 233
MsgBox ("El número de coches en la hora "
& R & " fue " & Numerodecoches(R))
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim Numerodecoches(23) As Integer
    Dim R As Integer

    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        R = 2
        Numerodecoches(R) = 233
        MsgBox("El número de coches en la hora " & R & " fue
" & Numerodecoches(R))
    End Sub
End Class
```

Nota: usamos Numerodecoches(23) en lugar de Numerodecoches(24) porque el número de elementos es uno más que el declarado debido a que se cuenta el cero. Así, si declaráramos Numerodecoches(24) tendríamos 25 elementos (correspondientes a los elementos del 1 al 24 más el elemento cero).

©2006-2029 aprenderaprogramar.com

El programa 2 será el siguiente. Recuerda también que usar una declaración tipo Dim A, B As Integer no sería válida porque supondría que A queda declarada como tipo Variant o tipo Object. Para que no haya duda usaremos Dim A%, B% ó Dim A As Integer, B As Integer

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com

Option Explicit
Dim Numerodecoches(23) As Integer
Dim A%, B As Integer
Dim R As Integer

Private Sub Form_Load()
A = 8
B = 4
R = A / B
Numerodecoches(R) = 233
MsgBox ("El número de coches en la hora "
& R & " fue " & Numerodecoches(R))
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
   Dim Numerodecoches(23) As Integer
   Dim A As Integer, B As Integer
   Dim R As Integer
   Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      A = 8
      B = 4
      R = A / B
      Numerodecoches(R) = 233
      MsgBox("El número de coches en la hora " & R & " fue
" & Numerodecoches(R))
   End Sub
End Class
```

Próxima entrega: CU00312A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

©2006-2029 aprenderaprogramar.com

ARRAYS MULTIDIMENSIONALES EN VISUAL BASIC

Tal y como explicamos en su momento, será posible crear arrays con más de una dimensión, pasando de la idea de lista, vector o matriz de una sola fila a la idea de matriz de m x n elementos, estructuras tridimensionales, tetradimensionales, etc. La sintaxis será:

```
Dim [Nombre del array]([dimensión1, dimensión2, ...]) As [Tipo de variable]
```

Si no se declara el tipo de variable el array será tipo Variant u Object (según la versión de Visual Basic que utilicemos) por defecto. Conviene tener cuidado con esto y seguir las mismas pautas que hemos indicado para las variables. La declaración de una matriz tradicional de m x n elementos podría ser:

```
Dim A(3, 2) As Integer
```

El número de elementos declarados se calcula teniendo en cuenta que la numeración de los índices empieza en cero, aunque ya hemos comentado que en algunas versiones existe una cláusula denominada Option Base que permite fijar como primer índice de los arrays el uno. En las versiones más recientes de Visual Basic (o con Option Base fijado en cero) tendremos una fila cero y una columna cero para la matriz declarada, con lo que el número de elementos será de 4 x 3 = 12. Decimos que hay 4 filas que son 0, 1, 2, 3 y tres columnas que son 0, 1, y 2. Si la versión de Visual Basic lo permite y fijamos Option Base en uno, la matriz será de 3 filas (1, 2 y 3) y dos columnas (1 y 2) siendo por tanto el número de elementos 3 x 2 = 6

Vamos a definir una matriz, array o arreglo de dos dimensiones a modo de ejemplo: queremos almacenar en una matriz el número de alumnos con que cuenta una academia ordenados en función del nivel y del idioma que se estudia. Tendremos 3 filas que representarán Nivel básico, medio o de perfeccionamiento y 4 columnas que representarán los idiomas (1 = Inglés, 2 = Francés, 3 = Alemán y 4 = Ruso). La declaración de dicha matriz sería:

```
Dim Alumnosfxniveleidioma(3, 4)
```

Podríamos asignar contenidos de la siguiente manera:

```
Alumnosfxniveleidioma(1, 1) = 7
Alumnosfxniveleidioma(1, 2) = 14
Alumnosfxniveleidioma(1, 3) = 8
Alumnosfxniveleidioma(1, 4) = 3
Alumnosfxniveleidioma(2, 1) = 6
Alumnosfxniveleidioma(2, 2) = 19
Alumnosfxniveleidioma(2, 3) = 7
Alumnosfxniveleidioma(2, 4) = 2
Alumnosfxniveleidioma(3, 1) = 3
Alumnosfxniveleidioma(3, 2) = 13
Alumnosfxniveleidioma(3, 3) = 4
Alumnosfxniveleidioma(3, 4) = 1
```

© 2006-2029 aprenderaprogramar.com

Los elementos de índice cero no los estamos utilizando, aunque podríamos hacerlo si quisiéramos.

La representación gráfica que podríamos asociar a esta asignación de datos sería esta matriz:

$$\begin{pmatrix} 7 & 14 & 8 & 3 \\ 6 & 19 & 7 & 2 \\ 3 & 13 & 4 & 1 \end{pmatrix}$$

La organización de la información en matrices nos generará importantes ventajas a la hora del tratamiento de datos en nuestros programas.

Para terminar en cuanto a multidimensionalidad, veamos casos de declaraciones con más de dos dimensiones. Para ello tomaremos como ejemplo el conteo del número de coches que pasan por una vía en un periodo de tiempo. La forma de declarar esos array sería la siguiente (hemos considerado que usamos el tipo de datos Integer para almacenar la información):

Duración del conteo	**Tipo de array**	**Declaración con *Visual Basic* (Nc es Númerodecoches)**
Un día	Array de un localizador (hora)	Dim Nc(24) As Integer
Varios días	Array de dos localizadores (hora y día)	Dim Nc(24, 31) As Integer
Varios meses	Array de tres localizadores (hora, día y mes)	Dim Nc(24, 31, 12) As Integer
Varios años	Array de cuatro localizadores (hora, día, mes y año)	Dim Nc(24, 31, 12, 2999) As Integer
Varios siglos	Array de cinco localizadores (hora, día, mes, año y siglo)	Dim Nc(24, 31, 12, 2999, 21) As Integer

Nota: en este ejemplo estamos considerando que no vamos a usar los índices cero de los arrays, pero podríamos hacerlo, de hecho es habitual.

© 2006-2029 aprenderaprogramar.com

Veamos lo que sería un ejemplo de programa con array multidimensional, en este caso usando un tipo String.

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Nombrealumno(5, 25) As String

Private Sub Form_Load()
Nombrealumno(3, 24) = "Pedro Hernández González"
MsgBox ("El alumno número 24 del curso tercero se llama " & Nombrealumno(3, 24))
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Dim Nombrealumno(5, 25) As String

   Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
      Nombrealumno(3, 24) = "Pedro Hernández González"
      MsgBox("El alumno número 24 del curso tercero se llama " & Nombrealumno(3, 24))
   End Sub
End Class
```

El resultado del programa es que se muestra el mensaje “El alumno número 24 del curso tercero se llama Pedro Hernández González”.

Próxima entrega: CU00313A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

ASISTENCIA DE VISUAL BASIC PARA ESCRITURA DE CÓDIGO

Ya hemos visto dónde debemos escribir el código asociado a un formulario. Hay que destacar que Visual Basic facilita el que podamos escribir código con rapidez a través de:

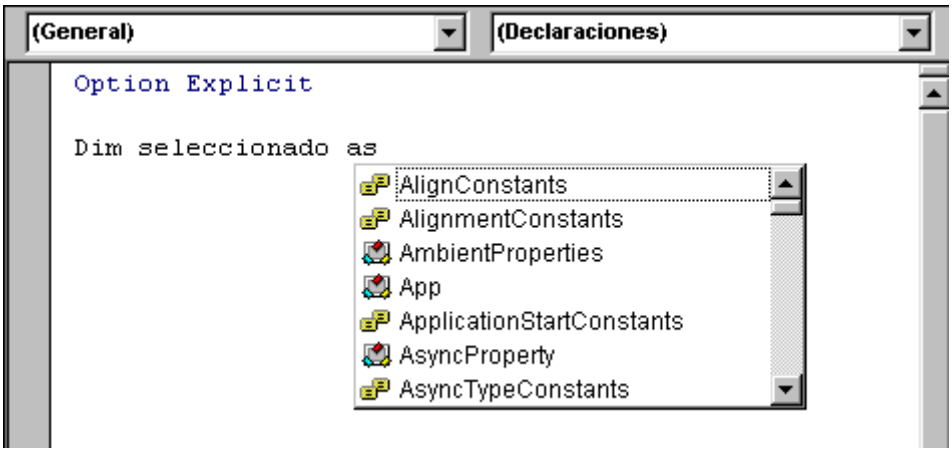
- Propuestas para elección de tipos de variable, propiedades de un objeto, etc.
- Corrección o indicación automática de errores que podamos cometer.
- Adelanto de la terminación del nombre de instrucciones o variables que hemos empezado a escribir.

Para que esta asistencia en la escritura esté disponible deberás comprobar:

En las versiones menos recientes, en el menú Herramientas - Opciones..., ventana Editor - Opciones del código), que tienes activadas las opciones correspondientes, siendo las principales: Comprobación automática de sintaxis (Habilitar sugerencias de corrección de errores), Lista de miembros automática, Información rápida automática y Sugerencias de datos automáticas.

En las versiones más recientes en el menú Herramientas, Opciones, activa la casilla “Mostrar todas las configuraciones”. A continuación elige Editor de texto, Basic, y comprueba que tienes activadas las opciones correspondientes, siendo las principales: General-Lista de miembros automática, General-Información de parámetros, Opciones Específicas – Inserción automática de construcciones End, Lista descriptiva (nuevo formato) de código, Habilitar sugerencias de corrección de errores y Habilitar resaltado de referencias y palabras clave.

Una vez verificado esto, vamos a comprobar cómo nos ayuda Visual Basic a escribir código. Dirígete a la ventana de código y escribe: Dim seleccionado As. Comprobarás cómo aparece una ventana donde existen diferentes posibilidades:



A través del cursor o del ratón, selecciona Boolean, con lo cual verás que se escribe automáticamente Boolean sin necesidad de teclearlo. Por tanto Visual Basic ha interpretado que ibas a definir el tipo de dato para la variable seleccionado y te ha mostrado las distintas posibilidades que tenías, facilitando la elección. Borra ahora la declaración de la variable y escribe lo siguiente:

```
Dim seleccionado as boolean
```

© 2006-2029 aprenderaprogramar.com

Pulsa enter y comprobarás que el texto automáticamente se convierte en:

```
Dim seleccionado As Boolean
```

Es decir, después de haber cometido un error al no poner correctamente las mayúsculas en as boolean, Visual Basic ha corregido automáticamente la escritura adaptándola a la sintaxis correcta. Esto es muy útil, ya que nos basta con tener las ideas claras respecto a lo que queremos escribir y saber su sintaxis aproximada. De los "detalles de sintaxis" se encargará Visual Basic, permitiendo que centremos nuestra atención en los aspectos principales de la programación.

Las correcciones no son (no pueden serlo) siempre automáticas. En algunos casos nos aparecerá un mensaje de error indicativo de que Visual Basic no entiende esa sintaxis y en otros no detectaremos que hemos cometido un error hasta tratar de ejecutar el programa.

Escribe por ejemplo: Dik seleccionado As Boolean

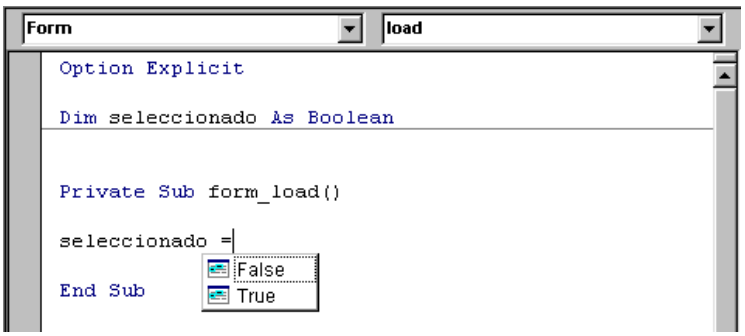
No habrá corrección automática pero al pulsar enter te aparecerá el texto remarcado o subrayado indicando que Visual Basic detecta que “hay algo extraño” en el código. Esto te servirá para prestar atención y darte cuenta de que en lugar de Dim has escrito Dik.

Si escribes Dim seleccionado As Voolean no hay corrección automática. Quizás aparezca un subrayado o remarcado indicando que se detecta algo extraño (o quizás no), ya que Visual Basic no sabe si tú has creado un tipo de variable al que has llamado Voolean. El error en este caso te saltará, si no lo corriges antes, al tratar de ejecutar el programa.

El editor de Visual Basic también nos puede ayudar a completar fragmentos de código obligatorios. Por ejemplo, en las versiones menos recientes cuando escribimos Private Sub form_load() y pulsamos enter nos aparece automáticamente una nueva línea End Sub, que debe acompañar obligatoriamente a la línea Private Sub form_load(). En las versiones más recientes si escribimos if seleccionado = True y pulsamos enter automáticamente nos aparecerá If seleccionado = True Then y en otra línea End If. Ya que son fragmentos de código que obligatoriamente tenemos que escribir, Visual Basic nos ahorra este trabajo. Aunque si lo preferimos, podemos desactivar ese autocompletado automático.

Si definimos seleccionado como una variable tipo Boolean y escribimos en el editor: seleccionado =

Comprobarás que aparece una ventanita con dos opciones: false y true.



Esto es así por estar la variable definida como tipo Boolean. Son los dos valores que se le podrían asignar y Visual Basic nos lo recuerda. Cuando estamos trabajando con decenas de variables es muy

© 2006-2029 aprenderaprogramar.com

útil. Podemos seleccionar False o True haciendo uso del cursor o del ratón, sin necesidad de escribirlo. Estas sugerencias automáticas podemos activarlas o desactivarlas según deseemos.

ORDENACIÓN DE LÍNEAS. NUMERACIÓN DE LÍNEAS. ETIQUETAS DE LÍNEAS.

Caso 1 versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
seleccionado = True
MsgBox ("El valor actual de seleccionado es " & seleccionado)
End Sub
```

Caso 1 versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        seleccionado = True
        MsgBox("El valor actual de seleccionado es " & seleccionado)
    End Sub
End Class
```

Caso 2 versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
10 seleccionado = True
20 MsgBox ("El valor actual de seleccionado es " & seleccionado)
End Sub
```

Caso 2 versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
10:     seleccionado = True
20:     MsgBox("El valor actual de seleccionado es " & seleccionado)
    End Sub
End Class
```

Caso 3 versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
Líneaasignacontenido: seleccionado = True
Líneamuestramensaje: MsgBox ("El valor actual de seleccionado es " & seleccionado)
End Sub
```

Caso 3 versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
Líneaasignacontenido: seleccionado = True
Líneamuestramensaje: MsgBox("El valor actual de seleccionado es " & seleccionado)
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Caso 4 versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
10 seleccionado = True
Líneamuestramensaje: MsgBox ("El valor actual de
seleccionado es " & seleccionado)
End Sub
```

Caso 4 versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
10:     seleccionado = True
Líneamuestramensaje: MsgBox("El valor actual
de seleccionado es " & seleccionado)
    End Sub
End Class
```

Los cuatro casos presentados son distintas posibilidades para numerar o etiquetar las líneas. Escribe el código en tu editor y compruébalo tú mismo.

El caso 1 corresponde a un programa en el que no existe identificación de las líneas, que para nosotros será lo habitual (en general es infrecuente identificar las líneas).

El caso 2 corresponde a un programa con las líneas numeradas (en las versiones más recientes después del número de línea se incluyen dos puntos). Recuerda a lo que era el Basic tradicional, para el que la numeración de líneas tenía una importancia notable al ser un tipo de programación altamente dependiente de la instrucción Go To (IrA), circunstancia que ya no se da con Visual Basic. Por tanto, en general no usaremos este tipo de numeración aunque Visual Basic la permite.

El caso 3 corresponde a un programa con las líneas identificadas por etiquetas. Cada línea tiene un nombre descriptivo que le precede, separado del contenido del código en sí por un signo de dos puntos. En circunstancias puntuales puede ser de interés, pero tampoco tiene sentido usarlo de forma constante para programar.

El caso 4 corresponde a un programa donde algunas líneas están numeradas, otras identificadas por una etiqueta y otras carecen de identificación.

Las posibilidades que se nos abren son amplias. Nosotros prescindiremos de la identificación de líneas, excepto para cuestiones puntuales como podría ser direccionar la gestión de errores. Sobre gestión de errores hablaremos más adelante.

Próxima entrega: CU00314A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

INSERCIÓN DE COMENTARIOS EN EL CÓDIGO

Existen distintas posibilidades para insertar comentarios en el código de Visual Basic como son:

La palabra clave Rem: la presencia de la palabra clave Rem (REM en mayúsculas algunas versiones) transforma en comentario todo lo que se encuentre a continuación de ella en una línea. Rem puede encontrarse al principio de una línea, con lo cual nada de lo que aparezca en dicha línea será ejecutado, o bien después de una o varias instrucciones que se encuentren en una línea. En algunas versiones no es necesario separador, mientras que en otras Rem debe estar separado de las instrucciones anteriores en la línea mediante dos puntos.

El signo apóstrofe ': la presencia del signo apóstrofe convierte en comentario todo lo que se encuentre detrás de él en una línea dada. Puede situarse al principio de una línea, con lo cual nada de lo que aparezca en dicha línea será ejecutado, o bien después de una o varias instrucciones sin necesidad de que existan dos puntos ni ningún otro signo de separación.

La parte del código que son comentarios es marcada por el editor de Visual Basic con un color verde que lo diferencia de las instrucciones que van a ser ejecutadas. En un mismo programa pueden aparecer indistintamente instrucciones Rem y apóstrofes. Veamos ejemplos equivalentes:

Ejemplo 1 (versiones menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
Rem A ejecutar cuando se carga el formulario
seleccionado = True
Rem Mostramos por pantalla mensaje
MsgBox ("El valor actual de seleccionado es " &
seleccionado)
End Sub
```

Ejemplo 1 (versiones más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
REM A ejecutar cuando se carga el formulario
        seleccionado = True
        REM Mostramos por pantalla mensaje
        MsgBox("El valor actual de seleccionado
es " & seleccionado)
    End Sub
End Class
```

Ejemplo 2 (versiones menos recientes):

```
'Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
'A ejecutar cuando se carga el formulario
seleccionado = True
'Mostramos por pantalla mensaje
MsgBox ("El valor actual de seleccionado es " &
seleccionado)
End Sub
```

Ejemplo 2 (versiones más recientes):

```
'Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
' A ejecutar cuando se carga el formulario
        seleccionado = True
        ' Mostramos por pantalla mensaje
        MsgBox("El valor actual de seleccionado
es " & seleccionado)
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Ejemplo 3 (versiones menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
seleccionado = True 'Ejecuta al cargar formulario
MsgBox ("El valor actual de seleccionado es " &
seleccionado) ' Mostramos por pantalla mensaje
End Sub
```

Ejemplo 3 (versiones más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
seleccionado = True 'Ejecuta al cargar formulario
MsgBox("El valor actual de seleccionado es " &
seleccionado) 'Mostramos por pantalla mensaje
    End Sub
End Class
```

Ejemplo 4 (versiones menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
seleccionado = True 'Ejecuta al cargar formulario
MsgBox ("El valor actual de seleccionado es " &
seleccionado) : Rem Mostrar por pantalla mensaje
End Sub
```

Ejemplo 4 (versiones más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
seleccionado = True REM Ejecuta al cargar formulario
MsgBox("El valor actual de seleccionado es " &
seleccionado) 'Mostramos por pantalla mensaje
    End Sub
End Class
```

El ejemplo 1 corresponde a un programa con comentarios insertados a través de Rem situados previos a la línea que comentan. El ejemplo 2 es igual con los Rem sustituidos por apóstrofes. En el ejemplo 3 los comentarios con apóstrofes se sitúan a continuación de las instrucciones, en la misma línea. Por último, el ejemplo 4 combina el uso de apóstrofes con Rem en la misma línea de instrucciones a ejecutar.

Los criterios para insertar comentarios pueden consultarse en el curso “Bases de la programación nivel I” de aprenderaprogramar.com. Los comentarios son una parte importante de los programas, por lo que es conveniente cuidar su estructuración y contenidos. Dado que con el pseudocódigo trabajamos con corchetes, podemos usar los corchetes detrás de los apóstrofes para "recordar" el carácter de comentario de dicho texto.

'[Esto es un comentario detrás de un apóstrofe. El corchete no es estrictamente necesario pero podemos ponerlo si queremos]

En las empresas de programación suele existir un manual de estilo donde se definen distintas reglas que deben cumplir los programadores, y la forma de comentar los programas suele ser una de ellas.

© 2006-2029 aprenderaprogramar.com

HERRAMIENTAS LÓGICAS Y MATEMÁTICAS EN VISUAL BASIC

Visual Basic clasifica los operadores con que trabaja de la siguiente manera:

- **Operadores aritméticos:** operadores utilizados para ejecutar cálculos matemáticos.
- **Operadores de comparación**: operadores utilizados para efectuar comparaciones.
- **Operadores de concatenación**: operadores utilizados para combinar cadenas de caracteres.
- **Operadores lógicos**: operadores usados para realizar operaciones lógicas.

Operadores Aritméticos	Operadores de comparación	Operaciones de concatenación	Operadores lógicos
Operador ^	< (Menor que)	&	Operador And
Operador *	<= (Menor o igual que)	+	Operador Eqv
Operador /	> (Mayor que)		Operador Imp
Operador \	>= (Mayor o igual que)		Operador Not
Operador Mod	= (Igual a)		Operador Or
Operador +	<> (Distinto de)		Operador Xor
Operador -	Is		
	Like		

La mayoría de estos operadores han sido explicados en el curso “Bases de la programación nivel I” de aprenderaprogramar.com. El operador & tiene interés para concatenar cadenas. Por ejemplo "Hoy puede" & "ser un gran día" equivale a "Hoy puede ser un gran día”. Te remitimos a la ayuda de Visual Basic para obtener mayor información sobre operadores. Ten en cuenta que puede haber pequeñas diferencias entre las distintas versiones de Visual Basic.

En cuanto a las funciones matemáticas, algunas de las funciones disponibles son las siguientes:

Funciones Matemáticas	Significado	Funciones Matemáticas	Significado
Abs	Valor absoluto	Log	Logaritmo neperiano
Atn ó Atan según versión	Arcotangente	Rnd	Generación de número aleatorio
Cos	Coseno	Sgn ó Sign según versión	Signo de un número
Exp	Exponenciación	Sin	Seno

© 2006-2029 aprenderaprogramar.com

Funciones Matemáticas	Significado	Funciones Matemáticas	Significado
Fix	Truncamiento tipo 1	Sqr ó Sqrt según versión	Raíz cuadrada
Int	Truncamiento tipo 2	Tan	Tangente

Las funciones relacionadas con ángulos trabajan en radianes. Por tanto si queremos operar con grados habremos de realizar la conversión oportuna.

Vamos a centrarnos en algunas funciones que no se han descrito en el curso “Bases de la programación nivel I”. Empezando por Fix e Int, se trata de funciones que devuelven la parte entera de un número, que es diferente a redondear. La diferencia entre ambas es que si el número es negativo Int devuelve el primer entero menor o igual mientras que Fix devuelve el primer entero mayor o igual.

Ejemplo:

$Int(8.7) = 8$ $\quad$ $Int(-8.7) = -9$

$Fix(8.7) = 8$ $\quad$ $Fix(-8.7) = -8$

La función *Rnd* permite generar números aleatorios. Por su importancia para la programación y para la verificación de programas, vamos a desarrollarla un poco más extensamente.

En las versiones más recientes de Visual Basic para usar una función matemática debemos escribir Math.nombreDeLaFuncion(valorACalcular). Por ejemplo, este es un pequeño programa para calcular la raíz cuadrada de un número.

Versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim num As Double

Private Sub Form_Load()
num = 1344
MsgBox ("La raíz cuadrada de " & num & " es " &
Sqr(num))
End Sub
```

Versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim num As Double

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        num = 1344
        MsgBox("La raíz cuadrada de " & num & " es " &
Math.Sqrt(num))
    End Sub
End Class
```

La función Sgn (Math.Sign en las versiones más recientes) devuelve el signo de un número o de una variable numérica tomando valor 1 para un número positivo, – 1 para un número negativo y cero cuando el número es cero. Supongamos A = 7.44, B = – 23, C = 0.

© 2006-2029 aprenderaprogramar.com

Versiones menos recientes:	*Sgn(A) = 1*	*Sgn(43) = 1*
	Sgn(B) = – 1	*Sgn(– 453.33) = – 1*
	Sgn(C) = 0	*Sgn(0) = 0*
Versiones más recientes:	*Math.Sign(A) = 1*	*Math.Sign(43) = 1*
	Math.Sign(B) = – 1	*Math.Sign(– 453.33) = – 1*
	Math.Sign(C) = 0	*Math.Sign(0) = 0*

Si vas a trabajar con constantes físicas o matemáticas te resultará de interés la instrucción Const para declaración de constantes. La ventaja de declarar una constante en vez de una variable es que la constante no puede variar en el transcurso del programa. Por tanto, se impide que por error pueda tener un valor no válido en un momento dado. Las constantes facilitan la documentación del programa y lo hacen fácil de modificar. Una declaración tipo de constante podría ser la siguiente:

```
Const Pi As Double = 3.14159265358979
```

El redondeo de un decimal a un entero se realiza en Visual Basic a través de la función CInt, correspondiente a un grupo de funciones denominadas "Funciones de conversión de tipos" que no vamos a entrar a estudiar. CInt(A) devuelve el valor de A redondeado al número entero más próximo. Prueba a visualizar el resultado de este código:

```
Dim A As Double
A = 7.32
MsgBox("El redondeo de 7.32 es " & CInt(A))
```

Próxima entrega: CU00315A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

GENERACIÓN DE NÚMEROS ALEATORIOS – FUNCIÓN Rnd

La generación de números aleatorios adquiere gran relevancia para un programador, pudiendo usarse para:

- Construcción preliminar de programas, en los que a falta de datos definitivos introducimos datos aleatorios.
- Simular procesos aleatorios (número resultante de tirar un dado, elección de un color por parte de una persona, número premiado en un sorteo de lotería, cantidad de personas que entran a un supermercado en una hora...)
- Verificación de programas, y en particular simular datos que le llegan al programa como si fueran datos provinientes de entradas de usuario, de una base de datos, de un fichero, etc.
- Otras aplicaciones.

Conviene recordar que "aleatorio" no puede confundirse con "cualquier cosa", "descontrol", "incierto", "impredecible", etc. Usaremos aleatorio más en el sentido de no predeterminado que en el de no predecible, ya que en general vamos a definir qué tipo de resultado queremos obtener y en qué rango de valores debe estar. Vamos a imaginar que Visual Basic genera números aleatorios como si fuera un robot lanzador de dardos muy preciso (robot Rnd). De este modo, cuando se le dice que comience a tirar dardos en distintas posiciones, repite siempre los lugares. Por ejemplo, si la diana está marcada con números, cada vez que le decimos que tire genera la misma secuencia: 7, 5, 6, 3, etc. ¿Cómo conseguir convertir este proceso predefinido en aleatorio? Pues simplemente poniendo a girar la diana (mayordomo Randomize) esta vez a una velocidad que depende del segundo del día en que nos encontremos. Así pues, el proceso lo dividimos en decirle al mayordomo que ponga a girar la diana y en decirle al robot que dispare. Bueno, un poco simple, pero ¿para qué complicarnos? Veamos la sintaxis a emplear:

1) Asignación de valor aleatorio a una variable

```
Randomize        [Instrucción que inicializa el generador de números aleatorios]
Variable = Rnd   [Obtención de un número aleatorio igual o mayor que cero y menor que uno]
```

En las versiones más recientes de Visual Basic la sintaxis es Randomize(): Variable = Rnd()

2) Mostrar un valor aleatorio.

```
Randomize: MsgBox (“El número aleatorio es “ & Rnd) para las versiones menos recientes
Randomize() : MsgBox("El número aleatorio es " & Rnd()) para las versiones más recientes
```

El valor devuelto por Rnd es de tipo Single, aunque si la variable a la que se asigna el contenido de Rnd es de otro tipo, tendrá lugar la conversión del valor originario para adaptarlo al tipo de variable de que se trate. Un aspecto importante a tener en cuenta es que el valor devuelto se encuentra en el rango 0 <= Rnd < 1. Es decir, el número devuelto puede ser cero pero no puede ser uno. Podemos trasladarnos a un intervalo distinto simplemente multiplicando. Por ejemplo Rnd * 100 ó Rnd() * 100 devolverá números comprendidos entre cero y cien excluido el cien. Si usamos CInt(Rnd * 100) ó CInt(Rnd() * 100)

© 2006-2029 aprenderaprogramar.com

estaremos convirtiendo el tipo Single que nos devuelve el Rnd en un tipo Integer, con lo que obtenemos valores enteros comprendidos entre 0 y 99.

Un caso muy habitual será el de querer generar números aleatorios comprendidos en un intervalo delimitado por un límite superior y un límite inferior siendo estos límites valores positivos. Para ello podemos usar la fórmula:

$$N = (Limsup - Liminf) * Rnd + Liminf$$

Limsup y Liminf son los límites superior e inferior respectivamente (valores positivos) y N el número aleatorio generado. El primer sumando puede adquirir el valor comprendido entre cero y el intervalo diferencia, excluido éste. El segundo sumando determina que el valor debe estar necesariamente comprendido entre el límite inferior y el límite superior.

Consideremos ahora que queremos generar valores enteros comprendidos entre a y b incluidos ambos, siendo a y b valores positivos. La fórmula a emplear sería:

$$N = Int((b - a + 1) * Rnd + a)$$

En este caso, usamos un truncamiento para garantizar que el valor superior del intervalo se encuentre entre los posibles valores de *N*.

Una instrucción Randomize te puede servir para varios Rnd. De todas formas, cuando tengas dudas, pon un Randomize asociado a cada uno de los Rnd.

Recuerda que en algunas versiones se usa Randomize y Rnd sin paréntesis y en otras se usa Randomize() y Rnd() con paréntesis.

Próxima entrega: CU00316A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

EJERCICIO CON RND Y RANDOMIZE

Vamos a generar un programa en Visual Basic donde utilizaremos el comando Rnd para generación de números aleatorios. El ejercicio es el siguiente: crear el código de un programa en el que se declaran tres variables tipo Single A, B y C cuyo valor se muestra en pantalla y debe estar comprendido entre cero y 10, excluido el diez.

Solución (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A!, B!, C As Single

Private Sub Form_Load()
Randomize
A = Rnd * 10
B = Rnd * 10
C = Rnd * 10
MsgBox ("A vale " & A & " ;B vale " & B & " ; C vale
" & C)
End Sub
```

Solución (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim A, B, C As Single

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Randomize()
        A = Rnd() * 10
        B = Rnd() * 10
        C = Rnd() * 10
        MsgBox("A vale " & A & " ;B vale " & B &
" ; C vale " & C)
    End Sub
End Class
```

EJERCICIO NÚMEROS ALEATORIOS

Crear el código de un programa que genera tres números enteros aleatorios *A*, *B*, *C* comprendidos entre 65 y 90, ambos inclusive, y los muestra en pantalla.

Solución (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A%, B%, C As Integer

Private Sub Form_Load()
Randomize
A = Int(Rnd * 26) + 65
B = Int(Rnd * 26) + 65
C = Int(Rnd * 26) + 65
MsgBox ("A vale " & A & " ;B vale " & B & " ; C vale
" & C)
End Sub
```

Solución (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim A, B, C As Integer

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Randomize()
        A = Int(Rnd() * 26) + 65
        B = Int(Rnd() * 26) + 65
        C = Int(Rnd() * 26) + 65
        MsgBox("A vale " & A & " ;B vale " & B &
" ; C vale " & C)
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Comentarios: Si quieres respetar la literalidad de la fórmula que expusimos en apartados anteriores del curso puedes escribir con igual resultado lo siguiente:

A = Int((90 - 65 + 1) * Rnd + 65); B = Int((90 - 65 + 1) * Rnd + 65); C = Int((90 - 65 + 1) * Rnd + 65)

CONCATENACIÓN DE ÓRDENES Y ORDENACIÓN SIMBÓLICA DEL CÓDIGO

La escritura de órdenes con Visual Basic se realiza con los criterios que ya hemos visto: las instrucciones pueden escribirse una debajo de otra en líneas independientes o bien en una misma línea separadas por dos puntos. Así estos dos códigos resultan equivalentes:

Código V1 (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A%, B%, C%, D%

Private Sub Form_Load()
A = 3: B = 2: C = A * B: D = C ^ 2
MsgBox ("D vale " & D)
End Sub
```

Código V1 (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim A, B, C, D%

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        A = 3 : B = 2 : C = A * B : D = C ^ 2
        MsgBox("D vale " & D)
    End Sub
End Class
```

Código V2 (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A%, B%, C%, D%

Private Sub Form_Load()
A = 3
B = 2
C = A * B
D = C ^ 2
MsgBox ("D vale " & D)
End Sub
```

Código V2 (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim A, B, C, D%

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        A = 3
        B = 2
        C = A * B
        D = C ^ 2
        MsgBox("D vale " & D)
    End Sub
End Class
```

La sangría al escribir código la estableceremos normalmente a través del tabulador, aunque para algunas instrucciones puede ser el propio editor de Visual Basic quien establezca la sangría automática. Si introducimos sangrías manualmente, el primer nivel de sangrado corresponderá a una primera pulsación del tabulador, el segundo nivel a dos pulsaciones y así sucesivamente...

© 2006-2029 aprenderaprogramar.com

A través del menú Opciones, Herramientas... (en algunas versiones hay que entrar además en Editor de texto, Basic, Tabulaciones) Visual Basic nos da la posibilidad de configurar el sangrado que deseamos a través de dos opciones:

1. **Sangría automática:** cuando incluyamos una tabulación en la primera línea de código todas las líneas siguientes comenzarán en esa posición de tabulación.

2. **Ancho de tabulación (tamaño de tabulación):** establece el ancho de la tabulación, ancho que puede oscilar entre 1 y 32 espacios, con un valor predeterminado de 4 espacios. Podemos establecer el valor que mejor se adapte a nuestras necesidades o gustos estéticos.

Una posibilidad que puede resultar interesante es la de usar un símbolo de prolongación de línea cuando tenemos una línea muy larga que no queremos separar a través de un símbolo de dos puntos. Para ello usamos el símbolo de concatenación de línea, que es un guión bajo (_). Dicho símbolo indicará que el texto que aparece en dos líneas, a efectos de ejecución del programa será tratado como si de una línea se tratara. Veamos el siguiente ejemplo:

Código (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A%, B%, C%, D%

Private Sub Form_Load()
A = 3
B = 2
C = A * B
D = C ^ 2
MsgBox ("Los valores de las variables que intervienen en
el proceso, tal y como " _
& "han sido definidas en el manual del programa son " _
& " A =" & A & " B =" & B & " C =" & C & " D =" & D)
End Sub
```

Código (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim A, B, C, D%

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        A = 3
        B = 2
        C = A * B
        D = C ^ 2
        MsgBox("Los valores de las variables
que intervienen en el proceso, tal y como " _
        & "han sido definidas en el manual
del programa son " _
        & " A =" & A & " B =" & B & " C =" &
C & " D =" & D)
    End Sub
End Class
```

Introducimos un símbolo de guión bajo cada vez que queramos continuar una línea. Hemos dividido una línea larga en tres renglones de longitud media. Los motivos para ello pueden ser varios, como estimarlo mejor para la ejecución del programa, o simplemente evitar que en el visor del editor nos quede la línea cortada y que podamos verla en toda su longitud. Prueba a escribir la línea dividiéndola en cuatro a través del guión bajo o dejándola completa sin divisiones. ¿No te parece que "el efecto" visual es muy distinto?

Próxima entrega: CU00317A

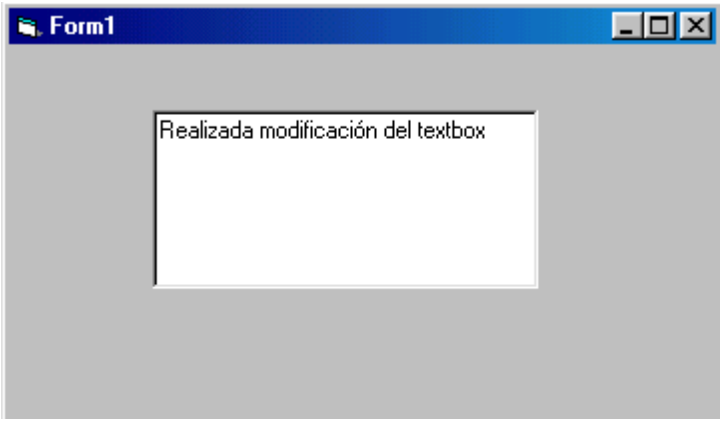
Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

INSTRUCCIÓN PEDIR CON VISUAL BASIC. LOS TEXTBOX.

La forma de pedir datos con Visual Basic resultará un tanto distinta a la que se utiliza cuando se trabaja con pseudocódigo puro o lenguajes estructurados como C. Una petición de datos no necesitará ser dispuesta en el código, sino que se obtendrá a través de un objeto prefabricado: un TextBox. Vamos a aprender a utilizarlo.

Visualiza el formulario y vete en el cuadro de herramientas al botón correspondiente a TextBox, representado por un símbolo ab|. Pulsando dos veces sobre él, o pulsando una vez y después insertándolo en el formulario, tendrás un TextBox cuyo nombre por defecto será Text1, TextBox1 ó similar. Agrándalo un poco y sin haber escrito código para el programa, da orden de ejecutar. El resultado es que te aparece el formulario con el TextBox sobre él. Puede que te aparezca vacío o que muestre un texto por defecto como Text1 dentro del TextBox. En la vista del diseño del formulario podemos escribir el mensaje que queramos dentro del TextBox. Borra el contenido de texto en el TextBox (si es que había alguno) y escribe por ejemplo "Realizada modificación del textbox" y pulsa enter. Quizás no hayas podido agrandar el TextBox en sentido vertical, ahora veremos cómo se puede hacer.



Un TextBox es un control que principalmente utilizamos para que el usuario introduzca datos, pero también puede mostrar una respuesta predeterminada o incluso servir para mostrar en vez de para pedir datos.

Un TextBox tiene una serie de propiedades, como puedes ver pulsando sobre el Textbox y recorriendo la lista que te aparece en la ventana Propiedades (si no tienes activada esta ventana, actívala). Prueba por ejemplo a cambiar el color de fondo (Backcolor). El mensaje inicial que aparecerá sobre el TextBox cuando comience el programa también es una propiedad: la propiedad Text. Prueba a cambiar esta propiedad (puedes tanto escribir un texto como dejar vacío el TextBox). Comprobarás que si escribes un contenido y pulsas enter, cuando ejecutas dicho texto aparece dentro del TextBox sobre el formulario. Te bastará con establecer Text vacío para que no haya ningún mensaje inicial cuando ejecutes el programa.

Muchas propiedades pueden establecerse en tiempo de diseño (cuando diseñamos el formulario) o en tiempo de ejecución (contenidos del código que se realizan cuando ejecutamos el programa). Escribe como texto para el TextBox en el diseño del formulario el texto "Text1". Escribe el siguiente código:

© 2006-2029 aprenderaprogramar.com

Código (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit

Private Sub Form_Load()
Text1.Text = ""
Text1.BackColor = vbGreen
End Sub
```

Código (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        TextBox1.Text = ""
        TextBox1.BackColor = Color.Green
    End Sub
End Class
```

El resultado es que el TextBox se muestra vacío y con color de fondo verde. El mostrarse vacío se debe a que hemos indicado que cuando se cargue el formulario la propiedad Text del control (un elemento como un TextBox se denomina "control") se establezca a cadena vacía (""). En versiones menos recientes de Visual Basic incluso se permitía escribir simplemente Text1 = "", ya que se consideraba que si no se especificaba propiedad era para referirnos a la propiedad Text. En las versiones más recientes es obligatorio indicar siempre la propiedad a modificar.

Hemos cambiado la propiedad de color indicándole que cambie a vbGreen ó Color.Green. VbGreen ó Color.Green se corresponden con constantes predefinidas por Visual Basic que tenemos disponibles para ser usadas cuando estemos escribiendo código.

Visual Basic define constantes para simplificar la programación. Puedes consultar "Constantes de *Visual Basic*" para conocer todas las constantes definidas, que se agrupan en constantes de color, Calendar, del compilador, de fecha, Dir, GetAttr y SetAttr, IMEStatus, Instr, StrComp, de código de teclas, MsgBox, QueryClose, Shell, StrConv, de color del sistema, VarType y otras.

Sólo citaremos algunas constantes que juzgamos interesantes. Una de ellas es vbCrLf, una constante que supone una indicación de salto de línea y retorno de carro. Cuando se introduce esta constante, equivale a una pulsación de enter. Su equivalente usando código ASCII es Chr(13) + Chr(10). Es decir, podemos escribir tanto una cosa como otra, resultando equivalentes. El interés de las constantes es que suelen resultar más sencillas de usar que su escritura con otro tipo de código. Prueba estos códigos y comprobarás que hacen lo mismo:

a) Caso 1

```
MsgBox ("22 salones" & vbCrLf & "4321 camarotes")
```

b) Caso 2

```
MsgBox ("22 salones" & Chr(13) & Chr(10) & "4321 camarotes")
```

© 2006-2029 aprenderaprogramar.com

En la ventana Propiedades puedes modificar la denominación del control (Text1 ó TextBox1) por otra denominación. Supón que en ese cuadro de texto se espera que el usuario introduzca una altura. Podemos establecer la propiedad Name como Textaltura. Así, podríamos escribir por ejemplo Textaltura.BackColor = vbGreen ó Textaltura.BackColor = Color.Green. También podríamos llamar al TextBox simplemente altura y escribir altura.BackColor = … , aunque este tipo de denominaciones son poco recomendables porque son menos claras. Conviene siempre respetar al menos un texto inicial que nos permita saber que esa denominación corresponde a un TextBox, ya que en caso contrario puede ser muy difícil de leer un programa. Por ejemplo, si vamos a pedir 4 datos que son nombre de edificio, longitud, anchura y altura podemos nombrar los TextBox como TextNomEdificio, TextLongitud, TextAnchura y TextAltura. Un mismo prefijo nos identifica todos los controles que son del tipo TextBox. Las normas para nombrar objetos deben seguir razonamientos similares a lo que hemos descrito para variables, tendentes a ser lo más claros y explicativos posibles.

Para terminar esta visión rápida de lo que son los TextBox, citaremos algunas propiedades que nos parecen interesantes:

- **TextAlign ó Alignment:** permite que el texto se alinee a la izquierda, a la derecha, o que aparezca centrado.
- **BorderStyle ó Appearance:** cambia la apariencia que puede ser tridimensional o plana.
- **Enabled:** si está establecido a True permite que se pueda escribir sobre el TextBox. Si está establecida a False, impide que el usuario pueda modificar el contenido del TextBox.
- **Maxlenght:** limita el número máximo de caracteres que puede admitir un TextBox. Si escribimos el valor 0, esto indica la ausencia de un máximo distinto del que imponga la memoria del sistema del usuario (lo cual será un valor muy grande, prácticamente equivalente a decir “se admite cualquier longitud de texto). Un valor mayor que 0 indica el número máximo de caracteres. Por ejemplo si establecemos el valor en 5, el usuario podrá escribir “perro” porque tiene 5 letras, pero no podrá escribir “serpiente” por tener más de 5 letras, sólo podría escribir << serpi >>
- **Multiline:** establecido a False supone que el TextBox consta de una línea única. Establecido a True permite que el TextBox conste de múltiples líneas. En algunas versiones es necesario establecer Multiline como true si queremos agrandar verticalmente el formulario.
- **Scrollbars:** permite añadir barras de desplazamiento verticales, horizontales o ambas al TextBox.
- **Visible:** determina si el TextBox resultará visible para el usuario o no.

Trata de modificar los valores de estas propiedades y observa los resultados. Haz lo mismo actuando en el código en vez de en el diseño del formulario. Podrás comprobar que existen muchas más propiedades además de las que hemos citado. Prueba a modificar algunas de ellas para comprender qué es lo que controlan.

Próxima entrega: CU00318A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

LA FUNCIÓN VAL

En Visual Basic una petición de datos al usuario no necesitará ser dispuesta en el código, sino que se obtendrá a través de un objeto prefabricado: un TextBox. Con frecuencia el dato que queremos obtener será un valor numérico, mientras que sin embargo el contenido de los TextBox es tipo texto. Por ello deberemos realizar una conversión. La función Val nos permite obtener el contenido de un TextBox en formato numérico (Integer, Single, Double, etc.).

El contenido que introduce un usuario a través de un TextBox tiene carácter de texto. Para convertir un número en formato texto a una variable numérica se usa la función Val. La sintaxis a emplear es:

```
Variable = Val(Cadena)
```

Ejemplo:

```
Dim A As Single
A = Val(TextBox1.Text)
```

Supone que la variable A, que ha sido definida como tipo Single, adopta el contenido numérico del TextBox denominado TextBox1.

La función Val deja de leer la cadena en el primer carácter que no puede reconocer como parte de un número. Los espacios en blanco, los tabuladores y los avances de línea se eliminan.

Val(" 38205 (Distrito Norte)") devuelve 38205 como valor numérico. Los espacios en blanco se eliminan, y el resto de cadena a partir de donde no se puede reconocer un valor numérico se ignora. Si la cadena empieza con un contenido no numérico Val devuelve cero. Por ejemplo Val (" (Distrito Norte) 38205 " devuelve cero.

La función Val sólo reconoce el punto (.) como separador decimal válido (aunque esto puede verse afectado por la configuración regional de tu ordenador). Cuando se usa un separador decimal diferente, debe realizarse un tratamiento previo de la cadena o utilizarse una función de conversión de tipos como CSng ó CDbl para convertir la cadena a un número.

Otra forma muy relevante de pedir datos con *Visual Basic* consiste en el uso de ventanas de petición de datos (Inputbox). Cuando se invoca la función Inputbox, se despliega una ventana independiente con un mensaje definido de solicitud de datos al usuario, que dispone de un Textbox asociado para responder y de botones para aceptar o cancelar. La sintaxis es del tipo n = Val(InputBox("Introduzca un número", "Introducción de número")), donde n es una variable que toma el valor que haya introducido el usuario. Valiéndote de la ayuda, prueba a crear distintos Inputbox.

A modo de ejemplo simple para ver el resultado de estas funciones, crea un proyecto y sitúa un TextBox sobre el formulario. Escribe el pequeño código que indicamos a continuación y comprueba sus resultados.

© 2006-2029 aprenderaprogramar.com

Código (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A As Single

Private Sub Form_Load()
A = Val(Text1.Text)
MsgBox ("A vale " & A)
A = Val(InputBox("Introduzca un número",
"Introducción de número"))
MsgBox ("A vale " & A)
End Sub
```

Código (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Public Class Form1
    Dim A As Single

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        A = Val(TextBox1.Text)
        MsgBox("A vale " & A)
        A = Val(InputBox("Introduzca un número", "Introducción de
número"))
        MsgBox("A vale " & A)
    End Sub
End Class
```

La función Val convierte a cualquier formato numérico, por tanto puede servir para introducir contenido en variables de tipo Integer, Long, Single, Double, etc.

Próxima entrega: CU00319A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

MOSTRAR CON VISUAL BASIC MEDIANTE LABELS

Ya sabemos que un Label sirve para insertar mensajes en un formulario que el usuario no podrá modificar. También hemos visto cómo insertarlos en el formulario. En cuanto a propiedades de los Labels, contenido, etc. su manejo es básicamente igual que el de los TextBox, con lo que prácticamente todo lo explicado en relación a los TextBox es también válido para los Labels. Por ejemplo, podrás escribir Label1.BackColor = Color.Green (ó Label1.BackColor = vbGreen según la versión de Visual Basic que estés utilizando) de forma análoga a como lo hicimos con TextBox.

El contenido que muestra el control se establece a través de la propiedad Text (en las versiones menos recientes propiedad Caption). Podemos escribir Label1.Text = "Introduzca aquí un entero positivo" (ó Label1.Caption = "Introduzca aquí un entero positivo" en otras versiones) para obtener ese mensaje dentro del Label1.

Aparte de las propiedades que ya conocemos podemos destacar:

- **Autosize:** si se establece como True, da lugar a que el tamaño del Label se ajuste al texto que muestra. Si quieres modificar el tamaño de un label en tiempo de diseño y no puedes posiblemente se deba a que Autosize está establecido como True.
- **Borderstyle:** permite cambiar la apariencia del borde del Label.
- **Font:** permite establecer tipo de letra, su tamaño, propiedades (cursiva, negrita, etc.)

Prueba a crear un Label y establecer diferentes mensajes y propiedades para el mismo.

Para mostrar un mensaje compuesto de distintas partes en un *Label* hemos de usar el símbolo de concatenación. Por ejemplo:

```
Label1.Text = msg1 & msg2    ó    Label1.Caption = msg1 & msg2  (según versión de VB en uso)
```

daría lugar a que se mostrara el contenido derivado de las variables msg1 y msg2. En las versiones menos recientes de Visual Basic se consideraba que si se escribiría Label = … nos referíamos al texto que debía mostrarse dentro del Label. Es decir, se permitía escribir:

```
Label1 = msg1 & msg2
```

Obteniendo el mismo resultado que en el caso anterior. En las versiones más recientes esto ya no se permite.

Vamos a trasladar el siguiente pseudocódigo a código usando Labels y TextBox:

```
Mostrar "Introduzca su peso en kgs por favor": Pedir Peso
```

a) A través del diseño del formulario.

© 2006-2029 aprenderaprogramar.com

Insertamos un Label en el formulario. Establecemos la propiedad Text (en otras versiones Caption) del Label como "Introduzca su peso en kgs por favor". Elegimos la propiedad Font como tamaño 10 y negrita. Creamos un TextBox llamado Text1 ó TextBox1, situado junto al mensaje.

b) A través del código (Recuerda que si escribes código donde citas un objeto como un TextBox, etc. dicho objeto debes haberlo creado previamente sobre el formulario en ver – objeto, y tienes que invocarlo respetando su nombre. Igualmente debes prestar atención al tamaño de los objetos para que los textos no se te corten, a la propiedad Visible, etc.)

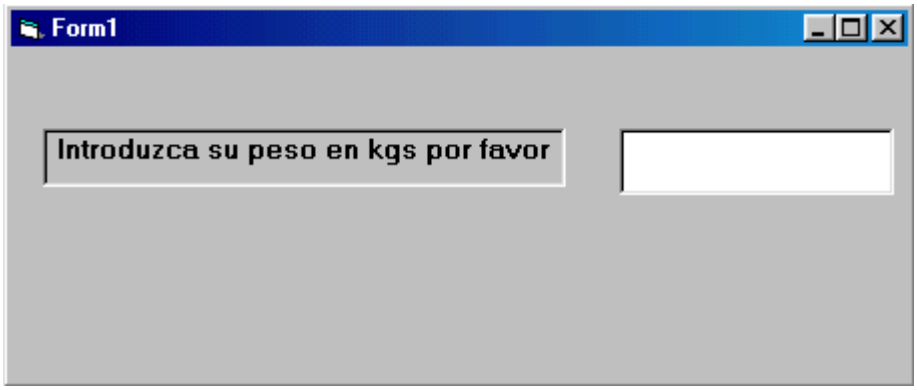
Código (versionesVB menos recientes):

```
Rem Curso Visual Basic
aprenderaprogramar.com
Option Explicit
Private Sub Form_Load()
Show
Label1.Caption = "Introduzca su peso en kgs
por favor"
Label1.Font.Size = 10
Label1.Font.Bold = True
End Sub
```

Código (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
Label1.Text = "Introduzca su peso en kgs por favor"
Me.Label1.Font = New Font("Arial", 10, FontStyle.Bold)
    End Sub

End Class
```

El resultado esperado es algo así:



Otra forma relevante de mostrar datos (también de pedirlos) es a través de ventanas de mensajes (MsgBox). Cuando se invoca la función MsgBox se despliega una ventana en la que se nos muestra un mensaje y una serie de botones a elegir entre Sí, No, Aceptar, Cancelar, Anular, Omitir, Reintentar...etc. La sintaxis es del tipo MsgBox("Valor no válido") ó Respuesta = MsgBox("¿Desea continuar?", vbYesNo), donde Respuesta es una variable tipo Integer donde se almacena un número asociado a la respuesta que ha dado el usuario. Por ejemplo, Respuesta valdrá 1 si el usuario ha elegido aceptar, valdrá 2 si ha elegido cancelar, valdrá 3 si ha elegido anular, etc.

Valiéndote de la Ayuda prueba a crear distintos MsgBox.

© 2006-2029 aprenderaprogramar.com

MOSTRAR CON VISUAL BASIC MEDIANTE PRINT

Hay distintos usos posibles de Print. En las versiones menos recientes existía una instrucción Print que permitía mostrar mensajes sobre un formulario y que es la que vamos a explicar a continuación. Si estás usando una versión reciente de Visual Basic esta instrucción no está disponible, por lo que puedes saltarte este apartado y pasar al siguiente. El uso de Print para mostrar mensajes sobre un formulario se basa en la escritura de texto entrecomillado y variables delimitadas por punto y coma. Ejemplo de este uso sería:

```
Print "La respuesta es"; A; "euros"
```

que daría lugar a un resultado del tipo: La respuesta es 32 euros. Utilizar una coma en lugar de un punto y coma es posible, pero la coma se traduce en la introducción de una tabulación. Así:

```
Print "La respuesta es", A; "euros"
```

da lugar a un resultado del tipo: La respuesta es 32 euros. También es posible usar el símbolo de concatenación &:

```
Print "La respuesta es " & A & " euros"
```

La diferencia respecto a usar puntos y comas estriba en que el punto y coma lleva a una introducción automática de un espacio antes y después de su ubicación, cosa que no ocurre con el símbolo &.

Podemos usar el símbolo de prolongación de línea para que dos líneas aparentes tengan consideración de una única línea:

```
Print "La respuesta es"; A; _
"euros"
```

Como hemos visto, se puede introducir una línea con un Print sin mensaje para dejar una línea vacía, o usar Tab para localizar el contenido a mostrar en un punto, por ejemplo:

```
Print Tab(18); A
```

La ventaja de Print es que es muy rápido de usar para visualizar contenidos y su inconveniente que a efectos estéticos es poco interesante, además de que ya no está disponible en las versiones recientes de Visual Basic. En general siempre será preferible usar otro tipo de formas de presentar datos como Labels o MsgBox.

Próxima entrega: CU00320A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

MATRICES O ARRAYS DE CONTROLES

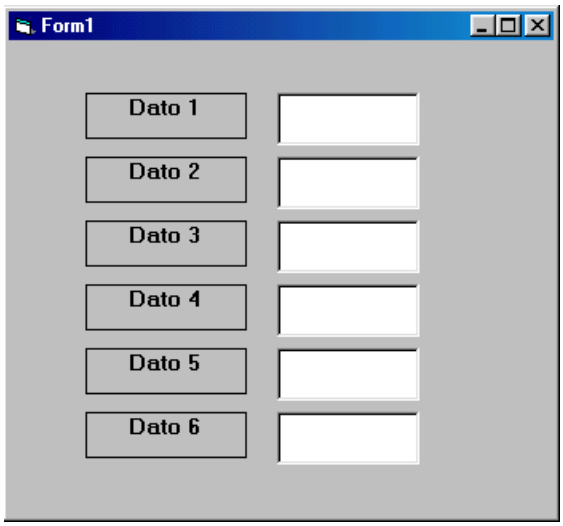
Ya hemos visto la utilidad del uso de arrays de variables. Por otro lado estamos viendo que los controles son objetos prefabricados, con algunas propiedades que recuerdan a las variables. Por ejemplo, el contenido de un TextBox "recuerda" al contenido de una variable. Siendo esto así, se comprenderá que igual que es interesante trabajar con arrays de variables puede ser interesante trabajar con arrays de controles.

CREAR UN ARRAY DE CONTROLES CON LAS VERSIONES MENOS RECIENTES DE VISUAL BASIC

Para crear un array de controles con las versiones menos recientes de Visual Basic haz lo siguiente:

- En primer lugar, crea el control deseado y mantenlo con el nombre que por defecto le ponga Visual Basic. Por ejemplo creamos un TextBox que se llamará Text1.
- A continuación, pulsa sobre el control y mediante Ctrl+C ó a través del menú Edición - Copiar, copia el control en el portapapeles.
- Mediante Ctrl+V ó a través del menú Edición - Pegar, pega el control sobre el formulario. Te aparecerá un mensaje del tipo: "Ya existe un control llamado Text1. ¿Desea crear una matriz de controles?" Este mensaje no te aparecerá en las versiones más recientes de Visual Basic. Explicaremos cómo proceder en el siguiente apartado.
- Responde sí y comprobarás que aparece un control sobre el formulario cuyo nombre es Text1(1), mientras que el control que ya existía anteriormente ha pasado a llamarse Text1(0).
- Si repites pegar, te seguirán apareciendo controles con nombre Text1(2), Text1(3), Text1(4), etc.

Esta posibilidad nos permite crear formularios del tipo:



En este formulario hemos definido 6 Labels denominados Label1(0), Label1(1), Label1(2), Label1(3), Label1(4) y Label1(5) así como 6 TextBox denominados Text1(0), Text1(1), Text1(2), Text1(3), Text1(4) y Text1(5). También podríamos haberlos nombrado simplemente como Label1, Label2, Label3, Label4,

© 2006-2029 aprenderaprogramar.com

Label5, Label6, etc. eligiendo en el momento de pegar no crear una matriz de controles. Pero al igual que ocurre con las variables, resultará mucho más cómodo de cara al tratamiento de datos el nombrarlas como una matriz de controles.

Podemos prescindir del control de localizador cero si así lo deseamos, simplemente borrándolo. De este modo en vez de una serie de seis números tipo 0, 1, 2, 3, 4, 5 nos quedaríamos con 1, 2, 3, 4, 5, 6.

CREAR UN ARRAY DE CONTROLES CON LAS VERSIONES MÁS RECIENTES DE VISUAL BASIC

Para crear un array de controles con las versiones más recientes de Visual Basic ha de recurrirse a usar código, lo cual resulta bastante más complicado que hacerlo de forma gráfica. Aquí vamos a poner la definición del código que habría que emplear para generar un formulario como el visto anteriormente, con 6 Labels y 6 TextBoxes. En este momento del curso no debemos entender este código por lo que no te entretengas en su estudio. Cuando hayamos avanzado un poco más y visto las instrucciones relativas a bucles, vuelve a este punto y repasa este código para comprobar que comprendes lo que hace. Nota: la referencia Me es hacia el propio formulario principal que estamos usando.

```
REM Curso Visual Basic aprenderaprogramar.com

Public Class Form1
    Dim LabelArray(6) As Label
    Dim TextBoxArray(6) As TextBox

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Dim i As Integer
        For i = 1 To 6
            LabelArray(i) = New Label : TextBoxArray(i) = New TextBox
            LabelArray(i).Text = "Dato " & i
            LabelArray(i).Width = 120 : TextBoxArray(i).Width = 100
            LabelArray(i).Height = 30 : TextBoxArray(i).Height = 30
            TextBoxArray(i).AutoSize = False
            LabelArray(i).TextAlign = ContentAlignment.MiddleCenter
            LabelArray(i).BorderStyle = BorderStyle.FixedSingle
            LabelArray(i).Font = New Font("Arial", 10, FontStyle.Bold)
            LabelArray(i).Location = New Point(100, i * 50)
            TextBoxArray(i).Location = New Point(260, i * 50)
            Me.Controls.Add(LabelArray(i)) : Me.Controls.Add(TextBoxArray(i))
        Next
    End Sub
End Class
```

Próxima entrega: CU00321A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

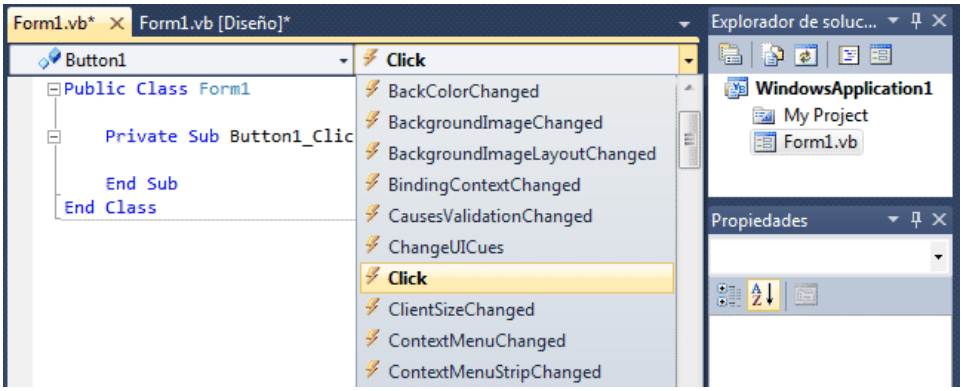
© 2006-2029 aprenderaprogramar.com

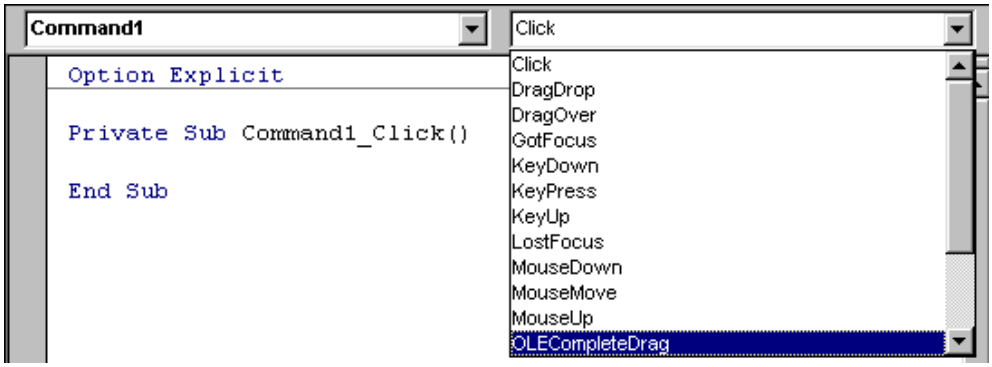
CÓDIGO ASOCIADO A EVENTOS. COMMAND BUTTONS.

La programación en Visual Basic es guiada por eventos. Por tanto, cuando pedimos un dato hemos de esperar a que suceda un evento para que continúe ejecutándose el código.

Sobre un formulario, coloca un TextBox, un Label y un Button (en algunas versiones de Visual Basic se denomina Commandbutton), que es otro de los elementos de los que disponemos en la barra de herramientas, cuyo símbolo es un rectángulo y que representa "un botón". Una vez añadidos estos tres controles al formulario vete a la ventana de código.

Observa que existen dos pestañas encima del espacio para la escritura de código. Si despliegas la pestaña de la izquierda, verás que se encuentran los distintos objetos que forman el programa y que por el momento son: el formulario o Form, la etiqueta o Label, la caja de texto o TextBox y el botón al que Visual Basic por defecto le pone nombre Button1 ó Command1 según la versión que estemos utilizando. Selecciona el Button1 (Command1). Ahora que tenemos en la pestaña izquierda seleccionado el botón, si pulsamos la pestaña derecha se muestra una lista de eventos que son detectables para el control que tenemos seleccionado. Por ejemplo, Click es el evento que se produce cuando el usuario hace click sobre el botón. MouseMove es el evento que se produce cuando el usuario mueve el puntero encima del CommandButton.





© 2006-2029 aprenderaprogramar.com

Cuando elegimos un objeto y un evento, aparece en la ventana de código el encabezado y terminación de un procedimiento de respuesta a dicho evento. Veamos un ejemplo. Agranda el objeto label sobre el formulario (establece la propiedad AutoSize a False para poder hacerlo), selecciona como control el botón en la pestaña izquierda y MouseMove como evento en la pestaña derecha y escribe lo siguiente:

Código (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit

Private Sub Command1_MouseMove(Button As
Integer, Shift As Integer, X As Single, Y As Single)
Label1.Caption = Label1.Caption & "Hola... "
End Sub
```

Código (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Button1_MouseMove(ByVal sender As Object,
ByVal e As System.Windows.Forms.MouseEventArgs)
Handles Button1.MouseMove
        Label1.Text = Label1.Text & "Hola... "
    End Sub
End Class
```

Ejecuta el programa y pasa el puntero del ratón sobre el botón. Comprobarás que aparecen múltiples "Hola…" sobre el formulario (en el label). Es decir, cada vez que tiene lugar el evento, se ejecuta el código asociado.

Crea ahora un procedimiento de respuesta al click de este tipo:

Código (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit

Private Sub Command1_Click()
Label1.Caption = Label1.Caption & "Hola... "
End Sub
```

Código (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Private Sub Button1_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Label1.Text = Label1.Text & "Hola... "
    End Sub
End Class
```

Comprobarás que cada vez que haces click sobre el botón se muestra el mensaje “Hola…” en el label sobre el formulario. Este tipo de procesos es vital en los programas: piensa cuando instalas un programa, cuántas veces pulsas Aceptar ó Continuar. Cada vez que lo haces, das lugar a que se ejecute un paquete de código. En cambio, si no lo haces, no tiene lugar el evento y puedes irte a tomar un café. Cuando vuelvas encontrarás que no ha sucedido nada.

Al igual que otros controles que venimos viendo, un botón tiene propiedades cuya gestión es igual a lo que hemos visto, en tiempo de diseño (trabajo gráfico sobre el formulario) o en tiempo de ejecución (a través de código). Podemos destacar las siguientes:

© 2006-2029 aprenderaprogramar.com

- **Name:** establece el nombre del botón. Recomendamos que siempre sea un prefijo común a todos los botones seguido de un término que nos permita identificar al botón. Por ejemplo, un botón para realizar un cálculo podríamos nombrarlo ButtonCalcular. Una rutina de respuesta a un click sería:

```
Private Sub ButtonCalcular_Click(…)

…

End Sub
```

- **Text ó Caption,** según la versión de Visual Basic que estemos empleando**:** establece el texto que se muestra al usuario sobre el botón. Por ejemplo, "Calcular".

- **AcceptButton (propiedad del formulario en versiones de Visual Basic más recientes) ó Default (propiedad del botón en versiones menos recientes):** es una propiedad interesante de los botones, ya que nos permite establecer si una pulsación de enter por parte del usuario equivale a un click a efectos de ejecución del código asociado. Es decir, supón que tienes un botón denominado ButtonCalcular y que además tienes un TextBox donde pides al usuario que introduzca un dato. Si establecemos en las propiedades del form como AcceptButton el ButtonCalcular (o establecemos Default como True para el botón en las versiones menos recientes), cuando el usuario pulse enter después de introducir el texto pedido, se ejecutará el código asociado. Prueba a establecer lo que hemos indicado con un formulario en el que hayas insertado un botón, un TextBox y un Label y comprueba que habiendo escrito un texto en el TextBox tras pulsar enter se ejecute el código previsto (que hará que se muestre “Has pulsado el botón o pulsado enter…”) :

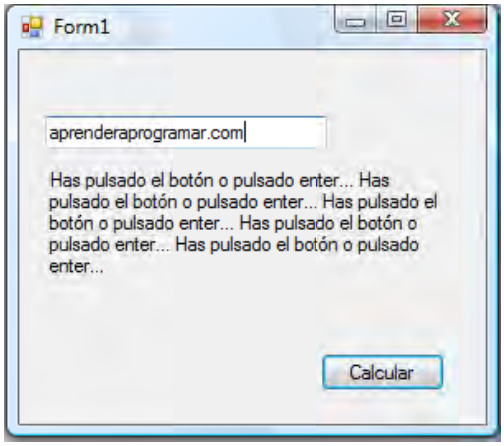
Código (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit

Private Sub Command1_Click()
Label1.Caption = Label1.Caption & "Has pulsado
el botón o pulsado enter... "
End Sub
```

Código (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Private Sub Button1_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Label1.Text = Label1.Text & "Has pulsado el
botón o pulsado enter... "
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com



Comprobarás que tiene efecto tanto hacer click sobre el Command Button como pulsar enter. Ten en cuenta que en muchas ocasiones, por ejemplo después de pedir un dato, una clave, etc. el usuario tiende a pulsar enter para dar paso a la continuación del programa. Esta propiedad te permite crear programas en los que se da relevancia a la pulsación de enter.

- **Otras propiedades:** al igual que con otros controles, existen diferentes propiedades que no hemos citado. Prueba a modificar y comprobar los efectos de algunas de ellas.

Próxima entrega: CU00322A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

EJERCICIOS CON MOSTRAR, PEDIR Y BOTONES EN VISUAL BASIC

EJERCICIO EJEMPLO PRIMERO

Transformar el algoritmo que se indica a continuación, relativo a calculo de necesidades de combustible, en un programa de Visual Basic.

ALGORITMO

1. **Inicio**
    2. [Definición de vehículos]
        3. Turismos = 32
        4. Todoterrenos = 11
    5. [Fin de definición de vehículos]
    6. [Definición de capacidad de depósitos]
        7. Capturismos = 40
        8. Captodot = 65
    9. [Fin de definición de capacidad de depósitos]
    10. [Cálculo de necesidades de combustible]
        11. Necesidadescom = Turismos * Capturismos + Todoterrenos * Captodot
    12. [Fin de cálculo de necesidades de combustible]
13. **Fin**

SOLUCIÓN EJERCICIO EJEMPLO PRIMERO

El código será algo como lo que se muestra a continuación. Antes de escribir el código tenemos que crear los objetos (controles) a que vamos a hacer referencia en el código, en concreto crearemos 6 labels, 4 TextBox y un botón.

© 2006-2029 aprenderaprogramar.com

Código (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Rem Declaración de variables
Dim Turismos%, Todoterrenos As Integer
Dim Capturismos!, Captodot As Single
Dim Necesidadescom As Single

Rem Contenido del formulario
Private Sub Form_Load()
Form1.Caption = "Cálculo de necesidades
combustible"
Label1(0) = "Por favor introduzca estos datos"
Label1(1) = "Número de turismos"
Label1(2) = "Número de todoterrenos"
Label1(3) = "Capacidad de turismos (litros)"
Label1(4) = "Capacidad de todoterrenos (litros)"
Command1.Caption = "Aceptar"
End Sub

Rem Cálculo y muestra resultados
Private Sub Command1_Click()
Label2.ForeColor = vbBlack
Label2.FontBold = True
Turismos = Val(Text1(0))
Todoterrenos = Val(Text1(1))
Capturismos = Val(Text1(2))
Captodot = Val(Text1(3))
Necesidadescom = Turismos * Capturismos +
Todoterrenos * Captodot
Label2 = "Las necesidades totales de combustible
son de " & Necesidadescom & " litros"
End Sub
```

Código (versiones VB más recientes):

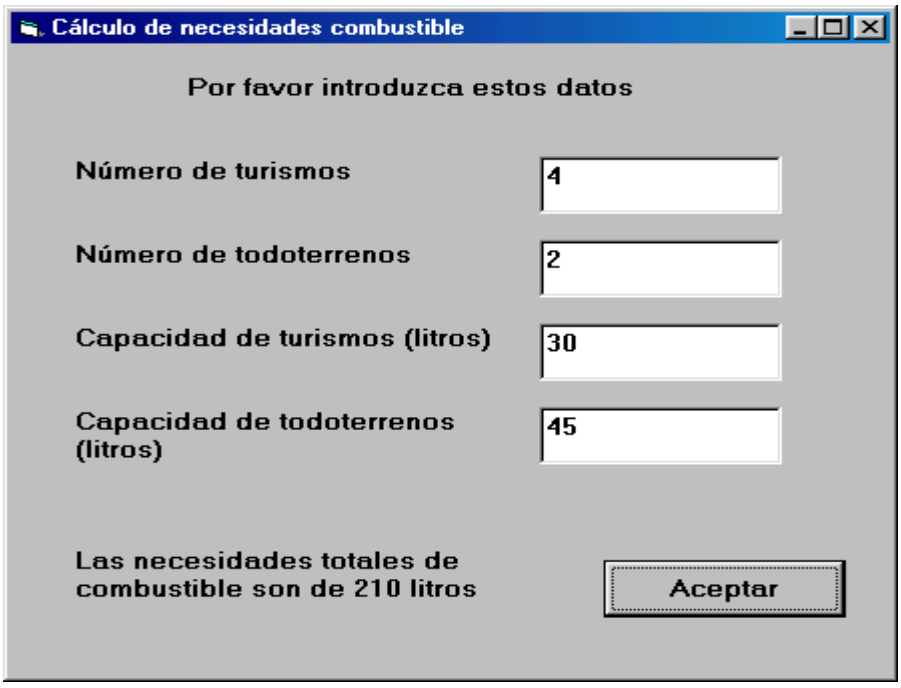
```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    REM Declaración de variables
    Dim Turismos, Todoterrenos As Integer
    Dim Capturismos, Captodot As Single
    Dim Necesidadescom As Single

    REM Contenido del formulario
    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
   Me.Text = "Cálculo de necesidades combustible"
   Label1.Text = "Por favor introduzca estos datos"
   Label2.Text = "Número de turismos"
   Label3.Text = "Número de todoterrenos"
   Label4.Text = "Capacidad de turismos (litros)"
   Label5.Text = "Capacidad de todoterrenos (litros)"
   Button1.Text = "Aceptar"
    End Sub

    REM Cálculo y muestra resultados
    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Label6.ForeColor = Color.Black
        Label6.Font = New Font("Arial", 10,
FontStyle.Bold)
        Turismos = Val(TextBox1.Text)
        Todoterrenos = Val(TextBox2.Text)
        Capturismos = Val(TextBox3.Text)
        Captodot = Val(TextBox4.Text)
        Necesidadescom = Turismos * Capturismos +
Todoterrenos * Captodot
        Label6.Text = "Las necesidades totales de
combustible son de " & Necesidadescom & " litros"
    End Sub
End Class
```

Téngase en cuenta que se han declarado como código diversas órdenes que podrían haberse introducido desde el diseño de objetos previo. Ten en cuenta también que todos los objetos que invocamos en el código los hemos creado previamente sobre el formulario (menú Ver – Objeto); los nombres de los objetos los pone Visual Basic (como en este ejemplo) al crearlos, o los establecemos en la ventana Propiedades nosotros mismos si deseamos cambiarlos. Si tratas de invocar un objeto no definido o con un nombre equivocado te saldrá un mensaje de error.

En un pequeño programa como este en el que tenemos varios labels, podemos optar tanto por usar arrays de controles (en este caso los controles tendrán nombres como Label1(0), Label1(1), Label1(2), etc.) o simplemente distintos controles (en este caso los nombres serán como Label1, Label2, Label3, etc.). Lo mismo podemos decir en relación a las cajas de texto (TextBox). Recordar que en las versiones menos recientes de Visual Basic es más sencillo crear arrays de controles que en las más recientes, ya que basta copiar y pegar un control y Visual Basic pregunta directamente si queremos crear el array.

El resultado gráfico después de introducir los datos y pulsar aceptar será algo así:

© 2006-2029 aprenderaprogramar.com



EJERCICIO EJEMPLO SEGUNDO

Transformar en código el pseudocódigo que se muestra a continuación, relativo a proporcionar el volumen de un cilindro dados su altura y diámetro.

1. **Inicio**
    2. Mostrar "Introduzca el diámetro, en metros" : Pedir D
    3. Mostrar "Introduzca la altura, en metros" : Pedir H
    4. R = D/2 : Pi = 3,141593
    5. V = Pi * (R ^ 2) * H
    6. Mostrar "El volumen del cilindro es de", V, "metros cúbicos"
7. **Fin**

SOLUCIÓN EJERCICIO EJEMPLO SEGUNDO

Creamos un formulario donde insertamos dos Labels (podemos usar arrays de controles como Label1(0) y Label1(1)) ó simplemente dos controles como Label1 y Label2, dos TextBox (podemos decir lo mismo que para los labels) y un Label para resultados al que vamos a ponerle nosotros el nombre y que llamaremos (LabelResultados), así como un botón al que llamaremos ButtonCalcular. El primer Label mostrará el mensaje de solicitud del diámetro y el segundo Label el correspondiente a la altura. A través de los TextBox el usuario introducirá los datos solicitados y a través del Label de resultados mostraremos los resultados. El código será algo así:

© 2006-2029 aprenderaprogramar.com

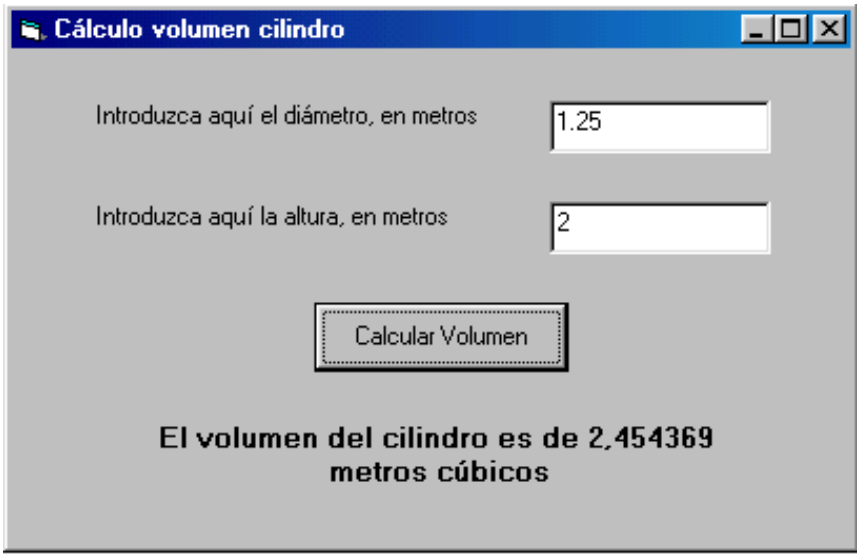
Código (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Rem Declaración de variables
Const Pi As Single = 3.141592654
Dim R!, H!, V As Single

Rem Contenido del formulario
Private Sub Form_Load()
Form1.Caption = "Cálculo volumen cilindro"
ButtonCalcular.Caption = "Calcular Volumen"
Label1.Caption = "Introduzca aquí el diámetro, en
metros"
Label2.Caption = "Introduzca aquí la altura, en
metros"
End Sub

Rem Cálculo y muestra resultados
Private Sub ButtonCalcular_Click()
R = Val(Text1) / 2
H = Val(Text2)
V = Pi * (R ^ 2) * H
LabelResultados.FontSize = 10
LabelResultados.FontBold = True
LabelResultados.Alignment = 2
LabelResultados = "El volumen del cilindro es de "
& V & " metros cúbicos"
End Sub
```

Código (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    REM Declaración de variables
    Const Pi As Single = 3.141592654
    Dim R, H, V As Single

    REM Contenido del formulario
    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
Me.Text = "Cálculo volumen cilindro"
ButtonCalcular.Text = "Calcular Volumen"
Label1.Text = "Introduzca aquí el diámetro, en metros"
Label2.Text = "Introduzca aquí la altura, en metros"
    End Sub

    REM Cálculo y muestra resultados
    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ButtonCalcular.Click
        R = Val(TextBox1.Text) / 2
        H = Val(TextBox2.Text)
        V = Pi * (R ^ 2) * H
        LabelResultados.Font = New Font("Arial", 10,
FontStyle.Bold)
        LabelResultados.TextAlign =
ContentAlignment.MiddleCenter
        LabelResultados.Text = "El volumen del
cilindro es de " & V & " metros cúbicos"
    End Sub
End Class
```

El resultado gráfico será similar a este:



© 2006-2029 aprenderaprogramar.com

EJERCICIO EJEMPLO TERCERO

Transformar en código el pseudocódigo que se indica a continuación, relativo a proporcionar el precio medio de un producto, calculado a partir del precio del mismo en tres establecimientos distintos.

1. **Inicio**
    2. Mostrar “Introduzca el precio del producto en el establecimiento número 1, en euros” : Pedir Precio1
    3. Mostrar “Introduzca el precio del producto en el establecimiento número 2, en euros” : Pedir Precio2
    4. Mostrar “Introduzca el precio del producto en el establecimiento número 3, en euros” : Pedir Precio3
    5. Media = (Precio1 + Precio2 + Precio3) / 3
    6. Mostrar “El precio medio del producto es”, Media, “Euros”
7. **Fin**

SOLUCIÓN EJERCICIO EJEMPLO TERCERO

En el formulario insertamos tres Labels y tres TextBox para petición de datos, así como un botón y un Label para cálculo y muestra de resultados. El código será algo así:

Código (versionesVB menos recientes):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Rem Declaración de variables
Dim Precio1!, Precio2!, Precio3!, Media As Single

Rem Contenido del formulario
Private Sub Form_Load()
Form1.Caption = "Cálculo precio medio producto"
ButtonCalcular.Caption = "Calcular precio medio"
Label1.Caption = "Introduzca aquí el precio en
establecimiento nº1, en euros"
Label2.Caption = "Introduzca aquí el precio en
establecimiento nº2, en euros"
Label3.Caption = "Introduzca aquí el precio en
establecimiento nº3, en euros"
End Sub

Rem Cálculo y muestra resultados
Private Sub ButtonCalcular_Click()
Precio1 = Val(Text1)
Precio2 = Val(Text2)
Precio3 = Val(Text3)
Media = (Precio1 + Precio2 + Precio3) / 3
LabelResultados.FontSize = 10
LabelResultados.FontBold = True
LabelResultados.Alignment = 2
LabelResultados = "El precio medio del producto
es de " & Media & " euros"
End Sub
```

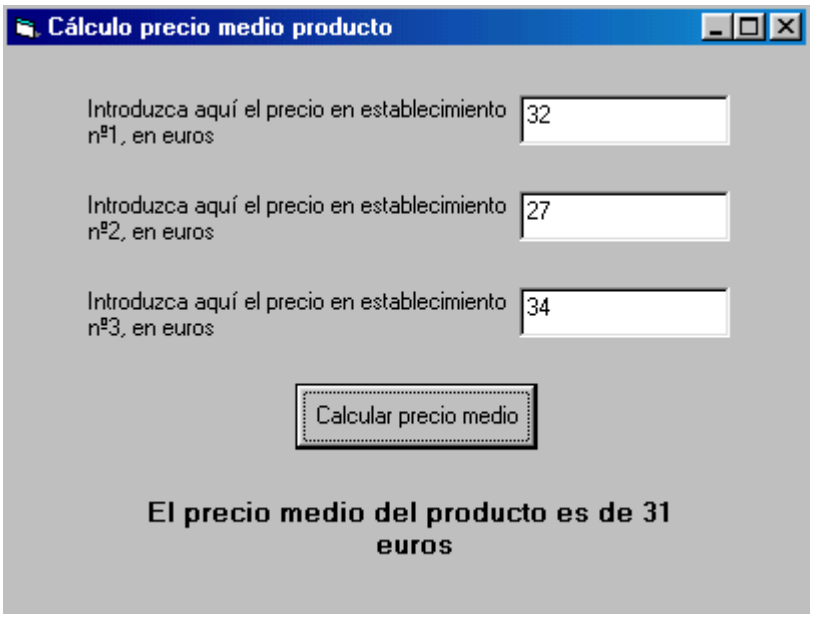
Código (versiones VB más recientes):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    REM Declaración de variables
    Dim Precio1, Precio2, Precio3, Media As Single

    REM Contenido del formulario
    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.Text = "Cálculo precio medio producto"
        ButtonCalcular.Text = "Calcular precio
medion"
        Label1.Text = "Introduzca aquí el precio en
establecimiento nº1, en euros"
        Label2.Text = "Introduzca aquí el precio en
establecimiento nº2, en euros"
        Label3.Text = "Introduzca aquí el precio en
establecimiento nº3, en euros"
    End Sub

    REM Cálculo y muestra resultados
    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ButtonCalcular.Click
        Precio1 = Val(TextBox1.Text)
        Precio2 = Val(TextBox2.Text)
        Precio3 = Val(TextBox3.Text)
        Media = (Precio1 + Precio2 + Precio3) / 3
        LabelResultados.Font = New Font("Arial", 10,
FontStyle.Bold)
        LabelResultados.TextAlign =
ContentAlignment.MiddleCenter
        LabelResultados.Text = "El precio medio del
producto es de " & Media & " euros"
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Para los controles podríamos haber usado arrays de controles si hubiéramos querido. Hemos usado controles independientes tipo Label1, Label2, Label3 y Text1, Text2 y Text3 en vez de arrays de controles. Si bien en líneas generales es ventajoso el uso de arrays cuando necesitamos automatizar procesos, para aplicaciones tan sencillas como ésta no tiene trascendencia hacerlo de una u otra manera.

El resultado gráfico será similar a este:



Próxima entrega: CU00323A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

IF … THEN (SI … ENTONCES) E IF … THEN … ELSE (SI … ENTONCES … SINO)

La sintaxis a emplear con Visual Basic es muy similar a la que se estudia en los cursos de algoritmia y pseudocódigo, con la sustitución del Si por If, del Entonces por Then, del SiNo por Else y del Finsi por End If. La única salvedad a tener en cuenta es que Visual Basic permite además de la escritura en bloques la escritura en una sola línea.

Sintaxis en bloque:

```
If [condición] Then
    Instrucción 1
    Instrucción 2
        .
        .
        .
    Instrucción n
Else
    Instrucción A
    Instrucción B
        .
        .
        .
    Instrucción Z
End If
```

Sintaxis en una sola línea:

```
If [condición] Then Instrucción 1 Else Instrucción A
```

También es posible encadenar varias acciones derivadas de un Then o de un Else en una sola línea. Sin embargo, recomendamos usar la opción de sintaxis en una sola línea exclusivamente para desarrollos sencillos como puede ser una instrucción en caso de que se cumpla la condición y otra en caso de que no se cumpla. Si intervienen varias instrucciones, siempre resultará más legible usar el formato de bloque.

La instrucción If ... Then es anidable dentro de sí misma, siendo siempre preferible el formato de bloque cuando se vayan a generar anidamientos.

Ejemplo:

```
If A > 7 And B < 5 Then
   MsgBox("A es mayor que 7 y B menor que 5")
End If
```

© 2006-2029 aprenderaprogramar.com

También válido es:

```
If A > 7 And B < 5 Then MsgBox ("A es mayor que 7 y B menor que 5")
```

Si introducimos una cláusula Else el modelo es:

```
If A > 7 And B < 5 Then
   MsgBox ("A es mayor que 7 y B menor que 5")
Else
   MsgBox ("A no es mayor que 7 ó B no es menor que 5")
End If
```

También válido usar el símbolo _ (guión bajo) para indicar continuación de línea:

```
If A > 7 And B < 5 Then MsgBox ("A es mayor que 7 y B menor que 5") _
Else MsgBox "A no es mayor que 7 ó B no es menor que 5"
```

Una cuestión a tener en cuenta es que cuando la condición indicada por un If en bloque no se cumple, se produce el salto a la siguiente instrucción inmediatamente posterior a la palabra End If. En cambio, cuando la condición indicada por un If en una línea no se cumple, se produce el salto a la siguiente línea ignorando todas las instrucciones que pudieran encontrarse en la misma línea del If. Veámoslo con un ejemplo. Ejecuta estos dos pequeños programas:

Programa 1 (versiones menos recientes VB):

```
Rem Curso Visual Basic aprenderaprogramar.com

Option Explicit
Dim a%

Private Sub Form_Load()
a = 3
If a = 5 Then
   MsgBox("Se cumple que a vale 5")
End If
a = 25
MsgBox("El valor actual de a es " & a)
End Sub
```

Programa 1 (versiones más recientes VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim a%

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        a = 3
        If a = 5 Then
            MsgBox("Se cumple que a vale 5")
        End If
        a = 25
        MsgBox("El valor actual de a es " & a)
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Programa 2 (versiones menos recientes VB):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim a%

Private Sub Form_Load()
a = 3
If a = 5 Then MsgBox ("Se cumple que a vale 5"): a = 25
MsgBox ("El valor actual de a es " & a)
End Sub
```

Programa 2 (versiones más recientes VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim a%

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        a = 3
        If a = 5 Then MsgBox("Se cumple que a
vale 5") : a = 25
        MsgBox("El valor actual de a es " & a)
    End Sub
End Class
```

En el programa 1 el resultado es que se muestra en pantalla "El valor actual de a es 25", mientras que en el programa 2 se muestra "El valor actual de a es 3". La razón para que en la versión 2 no se ejecute la orden a = 25 es que al no cumplirse la condición del If se produce un salto a la siguiente línea, ignorándose todas las instrucciones que se encuentren en la misma línea del If. En general consideramos recomendable habituarse a usar la sintaxis de bloque siempre pues resulta más fácil de interpretar y evita posibles confusiones como ésta, donde muchas personas podrían pensar que si no se cumple el if se ejecutará la instrucción que aparece en la misma línea a continuación.

Próxima entrega: CU00324A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

EJERCICIO EJEMPLO CON IF THEN ELSE EN VISUAL BASIC

Crear un programa que pida un número entero distinto de cero y nos muestre en pantalla un mensaje indicándonos si el número es par o impar. **Nota:** El procedimiento a emplear será basado en el uso del operador Mod.

SOLUCIÓN

Crearemos un Label de solicitud del número, un Textbox de entrada, un Button (Command Button en las versiones menos recientes de Visual Basic) y un Label para el cálculo y el resultado. Le ponemos los nombres y organizamos los controles en pantalla como nos parezca correcto (no tiene por qué coincidir con lo que ponemos aquí). Los nombres que hemos empleado nosotros son: LabelPideNum, TextNentero, ButtonHallar y LabelResultado.

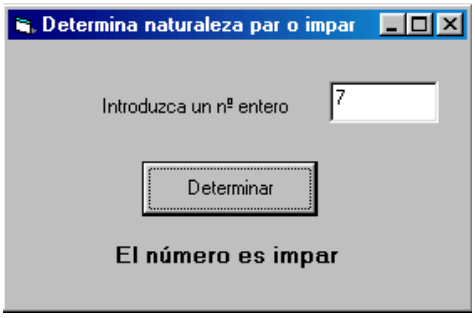
Código (versiones menos recientes VB):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Num As Integer
Dim Res As Integer
Private Sub Form_Load()
Form1.Caption = "Determina naturaleza par o
impar"
ButtonHallar.Caption = "Determinar"
LabelPideNum = "Introduzca un nº entero"
End Sub
Private Sub ButtonHallar_Click()
Num = Val(TextNentero)
Res = Num Mod 2
LabelResultado.FontSize = 10
LabelResultado.FontBold = True
If Res = 0 Then
   LabelResultado = "El número es par"
Else
   LabelResultado = "El número es impar"
End If
End Sub
```

Código (versiones más recientes VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim Num As Integer
    Dim Res As Integer
    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    Me.Text = "Determina naturaleza par o impar"
    ButtonHallar.Text = "Determinar"
    LabelPideNum.Text = "Introduzca un nº entero"
    End Sub

    Private Sub ButtonHallar_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles ButtonHallar.Click
        Num = Val(TextNentero.Text)
        Res = Num Mod 2
        LabelResultado.Font = New Font("Arial",
10, FontStyle.Bold)
       If Res = 0 Then
       LabelResultado.Text = "El número es par"
       Else
       LabelResultado.Text = "El número es impar"
        End If
    End Sub
End Class
```

Gráficamente:



Hemos escrito el If ... Then – Else como un bloque en vez de como una única línea. El motivo para ello es que aporta mayor claridad y facilidad de interpretación.

© 2006-2029 aprenderaprogramar.com

EJERCICIO ECUACIÓN DE SEGUNDO GRADO (ECUACIÓN CUADRÁTICA)

Crear un programa que resuelva la ecuación cuadrática tipo $ax^2 + bx + c$. El planteamiento del problema y su solución en forma de pseudocódigo se pueden encontrar en el curso "Bases de la programación nivel I" de aprenderaprogramar.com, dentro de los ejercicios relacionados con condicionales. Una vez creado el programa, hay que contrastar su correcto funcionamiento.

SOLUCIÓN

Usaremos tres Labels para pedir los parámetros a, b y c denominados Labela, Labelb y Labelc, y tres TextBox para que el usuario introduzca datos (Texta, Textb y Textc) además de un botón (ButtonCalcular). Nosotros hemos añadido además un mensaje de cabecera con un label (LabelTitulo). Mostraremos primero el código para versiones de Visual Basic más recientes y luego para versiones menos recientes. Ten en cuenta que la lógica de los programas es en ambos casos igual, lo único que varía son los detalles de la sintaxis.

CÓDIGO (PARA VERSIONES DE VISUAL BASIC MÁS RECIENTES)

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    REM Declaración de variables
    Dim aNumber, bNumber, cNumber, dNumber, eNumber As Single

    REM Contenido del formulario
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.Text = "Ecuación cuadrática"
        ButtonCalcular.Text = "Calcular x"
        LabelTítulo.Text = "Parámetros a, b y c para ecuación tipo a*x^2 + b*x + c = 0"

    End Sub

    REM Cálculo y muestra de resultados
    Private Sub ButtonHallar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ButtonCalcular.Click
        LabelResultado.Font = New Font("Arial", 10, FontStyle.Bold)
        aNumber = Val(Texta.Text)
        bNumber = Val(Textb.Text)
        cNumber = Val(Textc.Text)
        dNumber = bNumber ^ 2 - 4 * aNumber * cNumber
        eNumber = 2 * aNumber

        If dNumber = 0 Then
            LabelResultado.Text = "x1 = x2 = " & -bNumber / eNumber
        Else
            If dNumber > 0 Then
              LabelResultado.Text = "x1 = " & (-bNumber + Math.Sqrt(dNumber)) / eNumber
              LabelResultado.Text = LabelResultado.Text & vbCrLf & "x2 = " & (-bNumber -
Math.Sqrt(dNumber)) / eNumber
            Else
              LabelResultado.Text = "x1 = " & -bNumber / eNumber & " + " & Math.Sqrt(-dNumber) /
eNumber & " * " & Math.Sqrt(-dNumber) / eNumber & " * i"
            End If
        End If
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Nota: en versiones recientes de Visual Basic algunas letras están reservadas y no pueden ser usadas como nombres de variables. Por ejemplo la letra e. De ahí que en vez de nombrar a las variables como a, b, c, d, e las hayamos nombrado aNumber, bNumber, cNumber, dNumber y eNumber.

CÓDIGO (PARA VERSIONES DE VISUAL BASIC MENOS RECIENTES)

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Rem Declaración de variables
Dim a!, b!, c!, d!, e As Single

Rem Contenido del formulario
Private Sub Form_Load()
Form1.Caption = "Ecuación cuadrática"
CommandCalcular.Caption = "Calcular x"
LabelTítulo = "Parámetros a, b y c para ecuación tipo a*x^2 + b*x + c = 0"
Labela.Caption = "Valor parámetro a"
Labelb.Caption = "Valor parámetro b"
Labelc.Caption = "Valor parámetro c"
End Sub

Rem Cálculo y muestra de resultados
Private Sub CommandCalcular_Click()
LabelResultado.FontSize = 10
LabelResultado.FontBold = True
a = Val(Texta)
b = Val(Textb)
c = Val(Textc)
d = b ^ 2 - 4 * a * c
e = 2 * a
If d = 0 Then
   LabelResultado = "x1 = x2 = " & - b / e
Else
   If d > 0 Then
      LabelResultado = "x1 = " & (-b + Sqr(d)) / e
      LabelResultado = LabelResultado & vbCrLf & "x2 = " & (-b - Sqr(d)) / e
   Else
      LabelResultado = "x1 = " & -b / e & " + " & Sqr(-d) / e & " * i"
      LabelResultado = LabelResultado & vbCrLf & "x2 = " & -b / e & " - " & Sqr(-d) / e & " * i"
   End If
End If
End Sub
```

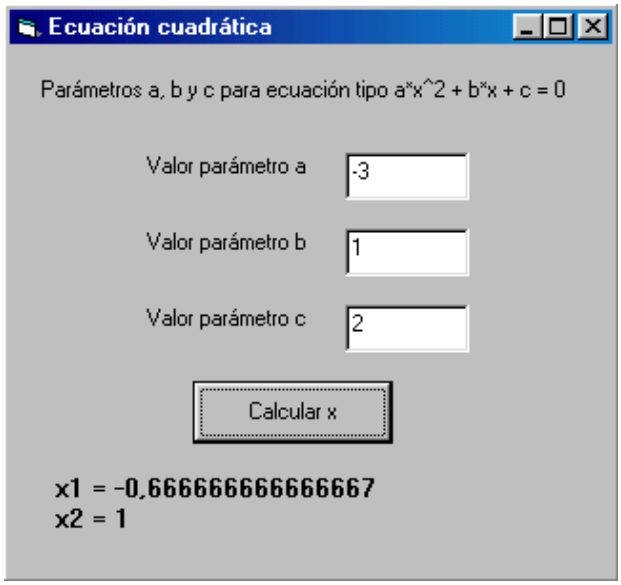
COMENTARIOS

El programa (ambas versiones de código) "funciona", si bien tiene sus limitaciones. Prueba a introducir los siguientes datos: a = 0, b = 3 y c = 1 . Te aparecerán resultados extraños (o un error como "Error 6 en tiempo de ejecución. Desbordamiento"). El motivo para esos resultados extraños o errores es que si a = 0 implica que e = 0 y al encontrarnos con una operación del tipo (– b + Sqr(d)) / e donde el denominador vale cero... ¡operación matemática no permitida! Esto ocurrirá mientras no preparemos el programa para detectar este tipo de situaciones y darles un tratamiento adecuado, cosa que veremos más adelante.

© 2006-2029 aprenderaprogramar.com

Estamos construyendo programas sencillos que en líneas generales responden bien cuando se dan las circunstancias previstas pero que fallan ante situaciones extrañas. Esto no debe preocuparnos por el momento, ya que a medida que progresemos como programadores iremos construyendo programas cada vez más sólidos.

Aspecto gráfico:



EJEMPLOS CON IF THEN ELSE EN VISUAL BASIC: PARADOJA DEL SI BURLADO POR UN INTERVALO

Reflexiona sobre el significado y prueba los siguientes códigos en tu ordenador:

Código 1 (versiones menos recientes de VB):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A%

Private Sub Form_Load()
A = 160
If 50 < A < 75 Then
   MsgBox ("Arabia")
Else
   MsgBox ("Eusebio")
End If
End Sub
```

Código 1 (versiones más recientes de VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim A As Integer

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        A = 160
        If 50 < A < 75 Then
            MsgBox("Arabia")
        Else
            MsgBox("Eusebio")
        End If
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Código 2 (versiones menos recientes de VB):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A%

Private Sub Form_Load()
A = 160
If A > 50 And A < 75 Then
   MsgBox ("Arabia")
Else
   MsgBox ("Eusebio")
End If

End Sub
```

Código 2 (versiones más recientes de VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim A As Integer

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        A = 160
        If A > 50 And A < 75 Then
            MsgBox("Arabia")
        Else
            MsgBox("Eusebio")
        End If
    End Sub
End Class
```

Si bien podría esperarse obtener el mismo resultado con ambos códigos, al ejecutar el código 1 obtenemos como resultado que se muestre en pantalla "Arabia", mientras que al ejecutar el código 2 se muestra "Eusebio". La razón: Visual Basic no admite comparaciones dobles del tipo x < y < z. Ante este tipo de escritura, sólo es evaluada la primera condición de las dos pretendidas. ¿Solución? No usar comparaciones dobles.

Este “resultado inesperado” puede ocasionar un buen quebradero de cabeza, ya que en un programa extenso no será nada fácil detectar este error. Como remedio, usaremos siempre la escritura de condiciones una detrás de otra y no de forma múltiple.

Próxima entrega: CU00325A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

INSTRUCCIÓN SEGÚN (CASO) HACER (SELECT CASE)

La sintaxis a emplear con Visual Basic es la siguiente:

```
Select Case [expresión]
        Case [valor expresión 1]
            Instrucción 1
            Instrucción 2
        Case [valor expresión 2]
            Instrucción 3
            Instrucción 4
                .
                .
                .
        Case [valor expresión n]
            Instrucción k
        Case Else
            Instrucción m
End Select
```

Select Case en Visual Basic equivale a la instrucción switch de otros lenguajes.

La expresión a evaluar puede ser un valor numérico o una cadena de texto. Sólo se puede evaluar una expresión y no múltiples expresiones. La evaluación de expresiones puede ser:

a) **De coincidencia:** por ejemplo, Case 12 indicaría que si la expresión evaluada vale 12 se ejecutarán las instrucciones anexas.
b) **De intervalo:** usando la palabra clave To. Por ejemplo, Case 12 To 14 indicaría que si la expresión evaluada tiene un valor comprendido entre 12 y 14 (incluidos los extremos de los intervalos), se ejecutarán las instrucciones anexas.
c) **De comparación:** usando la palabra clave Is. Por ejemplo, Case Is <= 14 indicaría que si la expresión evaluada tiene un valor menor o igual a 14 se ejecutarán las instrucciones anexas.

Cuando varios casos son válidos, sólo se ejecutan las instrucciones asociadas al primer caso verificado, siguiendo el flujo del programa por la siguiente instrucción después de End Select. Esto debemos tenerlo muy en cuenta, ya que si pretendemos que cuando se cumplan dos condiciones se ejecuten sendos bloques de instrucciones, convendrá hacerlo a través de la instrucción If en vez de a través de Select Case.

Las instrucciones Select Case se pueden anidar. Cada instrucción Select Case debe tener su correspondiente terminación End Select.

La instrucción Select Case presenta similitudes marcadas con la instrucción If - Then. Por ello, en general podremos conseguir resultados similares con ambas instrucciones y elegir usar una u otra dependerá de nuestro gusto y de cuál se adapte mejor a nuestros intereses, que en general serán ahorrar código y conseguir claridad para cumplir el objetivo. Otras veces será interesante anidar ambas instrucciones.

© 2006-2029 aprenderaprogramar.com

Prueba el siguiente código:

Código (versiones menos recientes VB):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A As Integer

Private Sub Form_Load()
A = 12
Select Case A
   Case 12
   MsgBox ("A vale 12")
   Case Is < 14
   MsgBox ("A es menor que 14")
   Case 160 To 200
   MsgBox ("A está comprendida entre 160 y 200")
   Case Else
      MsgBox ("No se ha verificado ninguna de las
condiciones previstas")
End Select
End Sub
```

Código (versiones más recientes VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Dim A As Integer
    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        A = 12
        Select Case A
            Case 12
                MsgBox("A vale 12")
            Case Is < 14
                MsgBox("A es menor que 14")
            Case 160 To 200
                MsgBox("A está comprendida entre
160 y 200")
            Case Else
                MsgBox("No se ha verificado
ninguna de las condiciones previstas")
        End Select
    End Sub
End Class
```

El resultado es que se muestra en pantalla "A vale 12". Sin embargo, a pesar de que la hipótesis A < 14 se cumple, no llega a evaluarse y por tanto las instrucciones asociadas a este caso no se ejecutan.

EJERCICIO

Transformar en código de Visual Basic el siguiente enunciado: crear un programa que pida al usuario un número de día y un número de mes, evalúe si la entrada es correcta y en caso de que así sea calcule el día del año que es, considerando un año no bisiesto.

SOLUCIÓN

Crearemos tres labels a los que denominaremos Label1, Label2 y LabelResultado. Label1 servirá para mostrar un mensaje solicitando que se introduzca el número de día, Label2 servirá para mostrar un mensaje solicitando que se introduzca el número de mes y LabelResultado servirá para mostrar el resultado.

Crearemos dos TextBoxes denominados Text1 y Text2. El primero de ellos servirá para que el usuario introduzca el dato de número de día y el segundo el dato de número de mes.

Finalmente tendremos un botón denominado ButtonCalcular que servirá para mostrar los resultados una vez el usuario haya introducido el dato de número de día y número de mes.

© 2006-2029 aprenderaprogramar.com

Código (versiones menos recientes VB):

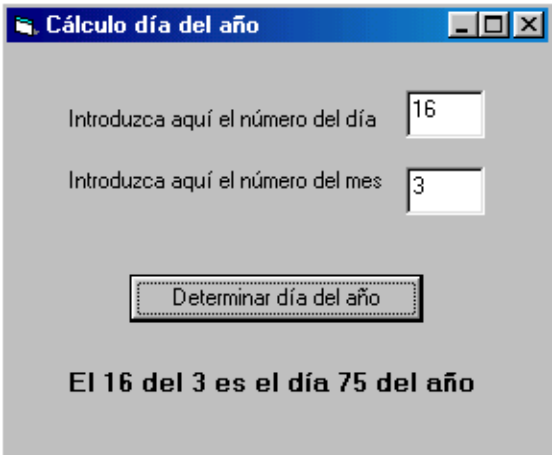
```
Rem Curso Visual Basic aprenderaprogramar.com
Rem [Cálculo del día del año]
Option Explicit
Dim Ndía%, Nmes%, DDA As Integer
Private Sub Form_Load()
Form1.Caption = "Cálculo día del año"
Label1.Caption = "Introduzca aquí el número del día"
Label2.Caption = "Introduzca aquí el número del mes"
ButtonCalcular.Caption = "Determinar día del año"
End Sub
Private Sub ButtonCalcular_Click()
Ndía = Val(Text1)
Nmes = Val(Text2)
LabelResultado.FontSize = 10
LabelResultado.FontBold = True
If Ndía >= 1 And Ndía <= 31 And Nmes >= 1 And Nmes <= 12 Then
  Select Case Nmes
    Case 1
      DDA = Ndía '[DDA es día del año]
    Case 2
      DDA = Ndía + 31
    Case 3
      DDA = Ndía + 59
    Case 4
      DDA = Ndía + 90
    Case 5
      DDA = Ndía + 120
    Case 6
      DDA = Ndía + 151
    Case 7
      DDA = Ndía + 181
    Case 8
      DDA = Ndía + 212
    Case 9
      DDA = Ndía + 243
    Case 10
      DDA = Ndía + 273
    Case 11
      DDA = Ndía + 304
    Case 12
      DDA = Ndía + 334
  End Select
  LabelResultado = "El " & Ndía & " del " & Nmes & " es el día " & DDA & " del año"
Else
  LabelResultado = "Datos proporcionados no válidos"
End If
End Sub
```

Código (versiones más recientes VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim Ndía, Nmes, DDA As Integer

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Me.Text = "Cálculo día del año"
        Label1.Text = "Introduzca aquí el número del día"
        Label2.Text = "Introduzca aquí el número del mes"
        ButtonCalcular.Text = "Determinar día del año"
    End Sub

    Private Sub ButtonCalcular_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonCalcular.Click
        Ndía = Val(Text1.Text)
        Nmes = Val(Text2.Text)
        LabelResultado.Font = New Font("Arial", 10, FontStyle.Bold)
        If Ndía >= 1 And Ndía <= 31 And Nmes >= 1 And Nmes <= 12 Then
            Select Case Nmes
                Case 1
                    DDA = Ndía '[DDA es día del año]
                Case 2
                    DDA = Ndía + 31
                Case 3
                    DDA = Ndía + 59
                Case 4
                    DDA = Ndía + 90
                Case 5
                    DDA = Ndía + 120
                Case 6
                    DDA = Ndía + 151
                Case 7
                    DDA = Ndía + 181
                Case 8
                    DDA = Ndía + 212
                Case 9
                    DDA = Ndía + 243
                Case 10
                    DDA = Ndía + 273
                Case 11
                    DDA = Ndía + 304
                Case 12
                    DDA = Ndía + 334
            End Select
            LabelResultado.Text = "El " & Ndía & " del " & Nmes & " es el día " & DDA & " del año"
        Else
            LabelResultado.Text = "Datos proporcionados no válidos"
        End If
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Aspecto gráfico:



Próxima entrega: CU00326A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

ESTRUCTURAS DE REPETICIÓN (BUCLES) CON VISUAL BASIC

Vamos a ver cómo materializar con Visual Basic estructuras de repetición que permitirán que en nuestros programas se realice un proceso n veces. En concreto veremos las instrucciones Desde … Siguiente (For … Next) con su cláusula Paso (Step), la instrucción Mientras … Hacer (Do While … Loop) y la instrucción Hacer … Repetir Mientras (Do … Loop While). Muchas veces podremos optar indistintamente por usar una instrucción u otra. En otros casos, es recomendable decantarse por una de ellas por hacer el programa más legible o sencillo que usando otras opciones.

INSTRUCCIÓN DESDE ... SIGUIENTE (FOR ... NEXT) Y CLÁUSULA PASO (STEP)

La sintaxis a emplear con Visual Basic es la siguiente:

```
For VAR = Vi  To Vf
        Instrucción 1
        Instrucción 2
                .
                .
                .
        Instrucción n
Next VAR
```

El incremento que sufre el contador es, por defecto, unitario. Es decir, el primer valor que toma VAR en el bucle será Vi, el segundo Vi + 1, el tercero (Vi + 1) + 1, etc. La modificación de dicho valor de incremento la realizaremos a través de la cláusula Step después del valor Vf. Step 2 implicará que en cada repetición del bucle el contador se incremente en dos unidades, Step 5 implicará que en cada repetición del bucle el contador se incremente en cinco unidades. Un paso negativo del tipo Step -1 supone que el contador decrece en vez de incrementarse. Si el paso es negativo, Vi necesariamente habrá de ser mayor que Vf, ya que en caso contrario no se producirá la entrada en el bucle.

Con este código se nos muestra en pantalla 3 veces hola (se ha invertido el sentido del bucle):

Código (versiones menos recientes VB):

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim VAR As Integer
Dim Vi As Integer
Dim Vf As Integer
Private Sub Form_Load()
Vi = 1
Vf = 3
For VAR = Vf To Vi Step –1
   '[También supondría tres repeticiones For
VAR = Vi to Vf]
   MsgBox ("hola")
Next VAR
End Sub
```

Código (versiones más recientes VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Dim VAR As Integer
        Dim Vi As Integer
        Dim Vf As Integer
        Vi = 1
        Vf = 3
        For VAR = Vf To Vi Step -1
            '[También supondría tres repeticiones
For VAR = Vi to Vf]
            MsgBox("hola")
        Next VAR
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Con Visual Basic resulta admisible usar Next sin indicar la variable que está sirviendo de guía del bucle, pues todo bucle ha de tener un cierre. Sin embargo, no lo creemos recomendable pues puede dificultar la lectura y depuración de los programas.

La instrucción For ... Next es anidable dentro de sí misma o dentro de otros tipos de bucles o estructuras, debiendo seguir las pautas que ya hemos comentado.

EJERCICIO

Transformar en código el siguiente planteamiento: queremos generar un programa que pida un número del 1 al 10 y nos muestre la "tabla de multiplicar" correspondiente.

SOLUCIÓN

Hemos realizado la pregunta relativa al número cuya tabla se desea conocer utilizando un InputBox. Igualmente válido hubiera sido hacerlo con un Label contenedor de la pregunta y un TextBox donde el usuario introduce la información. La tabla la mostramos sobre un Label denominado LabelTabla.

Código (versiones menos recientes VB):

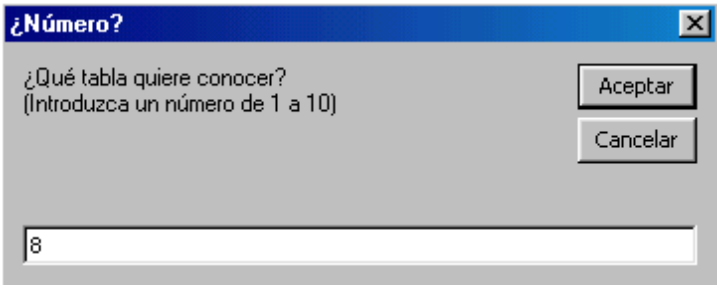
```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim T As Integer
Dim i As Integer

Private Sub Form_Load()
Form1.Caption = "Tabla"
T = Val(InputBox("¿Qué tabla quiere conocer?" & vbCrLf & _
"(Introduzca un número de 1 a 10)", "¿Número?"))
LabelTabla.FontBold = True
LabelTabla.Alignment = 2
LabelTabla = "TABLA DEL " & T & vbCrLf & vbCrLf

For i = 1 To 10
   LabelTabla = LabelTabla & T & " * " & i & " = " & T * i & vbCrLf
Next i
End Sub
```

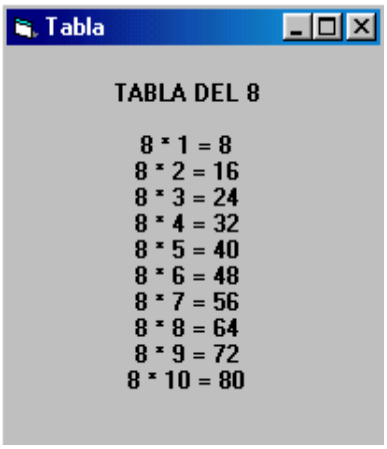
Código (versiones más recientes VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim T As Integer
    Dim i As Integer

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Me.Text = "Tabla"
        T = Val(InputBox("¿Qué tabla quiere
conocer?" & vbCrLf & _
"(Introduzca un número de 1 a 10)", "¿Número?"))
LabelTabla.Font = New Font("Arial", 10,
FontStyle.Bold)
LabelTabla.TextAlign = ContentAlignment.MiddleCenter
LabelTabla.Text = "TABLA DEL " & T & vbCrLf &
vbCrLf
        For i = 1 To 10
          LabelTabla.Text = LabelTabla.Text & T &
" * " & i & " = " & T * i & vbCrLf
        Next i
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Gráficamente:





EJERCICIO

Transformar en código el siguiente planteamiento: queremos generar un programa que muestre todos los enteros comprendidos entre un número definido por el usuario y 100.

SOLUCIÓN

Código (versiones menos recientes VB):

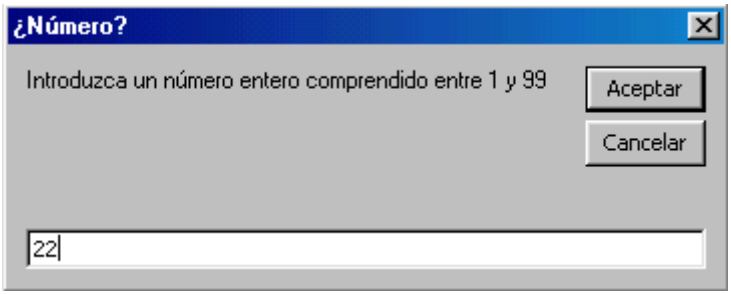
```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Num As Integer
Dim i As Integer

Private Sub Form_Load()
Form1.Caption = "Enteros"
Num = Val(InputBox("Introduzca un número
entero comprendido entre 1 y 99", "¿Número?"))
LabelTabla.Alignment = 2
LabelTabla.FontBold = True
LabelTabla = "ENTEROS ENTRE " & Num & " y
100" & vbCrLf & vbCrLf
For i = Num To 100
   LabelTabla = LabelTabla & i & " "
Next i
End Sub
```

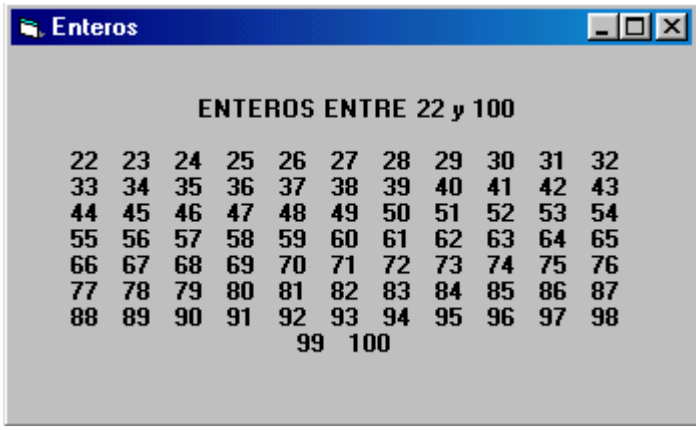
Código (versiones más recientes VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Dim Num As Integer
        Dim i As Integer
        Me.Text = "Enteros"
        Num = Val(InputBox("Introduzca un número
entero comprendido entre 1 y 99", "¿Número?"))
        LabelTabla.TextAlign =
ContentAlignment.MiddleCenter
        LabelTabla.Font = New Font("Arial", 10,
FontStyle.Bold)
        LabelTabla.Text = "ENTEROS ENTRE " & Num
& " y 100" & vbCrLf & vbCrLf
        For i = Num To 100
            LabelTabla.Text = LabelTabla.Text & i
& " "
        Next i
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Gráficamente:





INSTRUCCIÓN MIENTRAS ... HACER (DO WHILE ... LOOP)

La sintaxis que hemos de seguir es la siguiente:

```
Do While [condición]
    Instrucción 1
    Instrucción 2
        .
        .
        .
    Instrucción n
Loop
```

Las instrucciones tipo Do While son anidables dentro de sí mismas o dentro de otras estructuras. Es importante verificar que los bucles diseñados con esta instrucción dispongan de una condición de salida válida.

Ejemplo: k = 0: Do while k < 5. Dentro del bucle tendremos que incrementar el valor de k, k = k +1.

© 2006-2029 aprenderaprogramar.com

INSTRUCCIÓN HACER ... REPETIR MIENTRAS (DO ... LOOP WHILE)

La sintaxis a utilizar es:

```
Do
    Instrucción 1
    Instrucción 2
        .
        .
        .
    Instrucción n
Loop While [condición]
```

Un Do ... Loop While es anidable dentro de sí mismo o dentro de otras estructuras. Es importante verificar que los bucles diseñados con esta instrucción dispongan de una condición de salida válida.

Ejemplo:

```
Do
    LabelTabla.Text = LabelTabla.Text & "Iteración " & k & vbCrLf
    k = k + 1
Loop While k <= 5
```

Visual Basic admite además de la cláusula While, usar el término Until, como equivalente a "hasta que se cumpla que". Así Loop Until i >= 3 significaría "Repetir hasta que i sea mayor o igual que 3". En un bucle en el que i parte de un valor cero y se incrementa unitariamente Do While i < 3 sería equivalente a Do Until i >= 3, y Loop Until i >= 3 sería equivalente a Loop While i < 3. Dado que podemos valernos de equivalencias, puede evitar confundirnos el usar preferentemente un mismo tipo de expresión, sabiendo que disponemos de otra equivalente.

Próxima entrega: CU00327A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

INSTRUCCIÓN LEER E INSTRUCCIÓN GUARDAR. INTRODUCCIÓN A ARCHIVOS CON VB.

> **Advertencia:** una manipulación incorrecta de ficheros puede dar lugar a la pérdida de datos, al malfuncionamiento de programas o a fallos del sistema operativo del ordenador. Es necesario disponer de copias de seguridad de toda información que se considere importante.

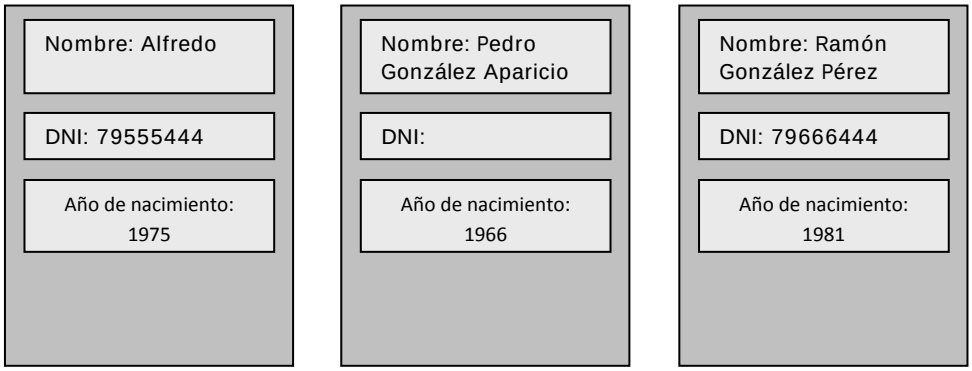
Guardar datos a un disco o recuperar los datos previamente guardados son procesos fundamentales en cualquier lenguaje informático. La importancia del almacenamiento de contenidos es obvia: envío de información a otros usuarios, posponer el trabajo varios días o semanas sin tener que introducir manualmente los datos de nuevo, acceso a información almacenada en sistemas remotos, etc. Incluso para desarrollos de software de relativamente corta longitud resulta relevante la gestión de datos, por ahorrar una cantidad de tiempo considerable.

Ficheros y bases de datos constituyen un cuerpo de conocimiento de cierta complejidad y extensión. Por motivos de espacio no vamos a hacer más que una breve introducción a ellos.

Podemos pensar en los ficheros o archivos de forma análoga a lo que sería un archivo físico: un lugar donde hay información almacenada. Ahora bien, eso no nos dice cómo se encuentra esa información: si está ordenada (alfabéticamente, de mayor a menor, en impresos con campos predefinidos), si son datos de un tipo o de varios tipos (numéricos, tipo texto, mezcla de tipos de datos...), si se leen usando alguna clave de interpretación (una partitura no se lee igual que un texto), etc. Todo esto es relevante y por ello normalmente sabremos qué tipo de datos contiene el fichero y cómo están ordenados antes de acceder a él. No es lo mismo guardar "*3246*" como número tipo *Integer* que como cadena de caracteres. Un tipo *Integer* ocupa 2 *bytes* y tiene una "clave" para su lectura, mientras que como caracteres cada cifra ocupa un *byte* y se lee como si de texto se tratara.

Un asunto a considerar es si la información está contenida en campos de longitud predefinida. Consideremos un archivo donde se almacenan los nombres, números de DNI y año de nacimiento de 3 personas. Podemos considerar diferentes casos:

a) Existen campos de longitud predefinida, que pueden carecer de información por no estar disponible.



En el fichero físico la información la guardamos en hojas preimpresas donde cada apartado tiene una longitud disponible para escribir en él. En el fichero informático la información la guardamos en

© 2006-2029 aprenderaprogramar.com

registros (un registro equivale a una hoja del fichero físico), constando cada registro de 3 campos (campo "Nombre", campo "DNI" y campo "Año de nacimiento"). Los campos serían por ejemplo:

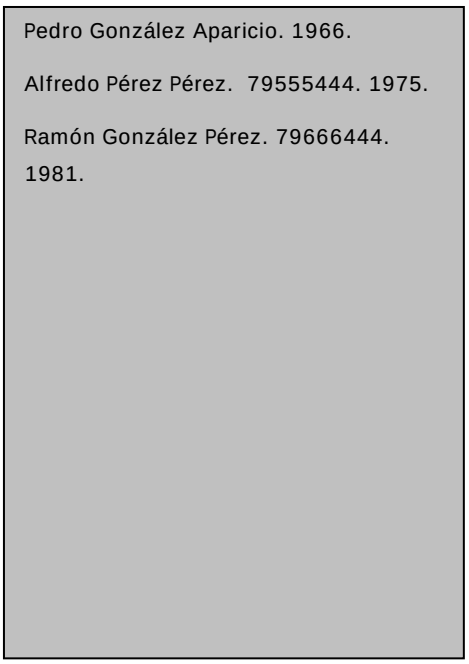
Nombre: tipo texto 25 caracteres de longitud.

DNI: tipo entero largo (*Long*).

Año de nacimiento: tipo entero (*Integer*).

Cada registro ocupa *25 + 4 +2 = 29 bytes*. Si tenemos tres registros el fichero ocupa *29 * 3 = 87 bytes*, independientemente de que los datos realmente existan o no, ya que el espacio se encuentra disponible para albergar un dato.

b) La información no está en campos de longitud predefinida.

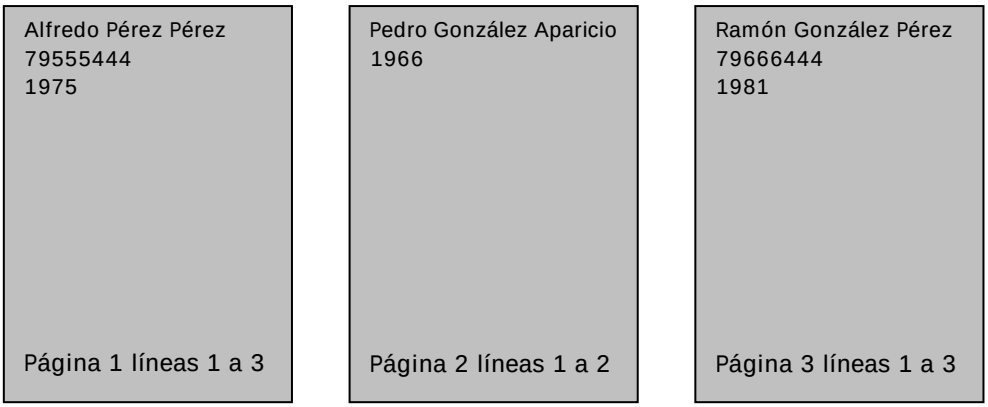


En el fichero físico la información la guardamos en hojas donde escribimos los datos disponibles para cada persona. En el fichero informático la información la guardamos en líneas (cada línea equivale a la información disponible de una persona) sabiendo que cada línea puede tener información de uno o varios aspectos (nombre, DNI y edad) y una longitud variable.

No sabemos cuántos *bytes* nos ocupa cada línea ni cuántos *bytes* nos ocupa el fichero. No sabemos en qué posición se encuentra una información dada porque no tenemos referencias de situación. Ante ello, optamos por hacer una lectura tipo "texto" en la que un número de 8 cifras nos va a requerir *8 bytes* frente a los *4 bytes* que supondrían como tipo *Long.*

© 2006-2029 aprenderaprogramar.com

c) La información no está en campos de longitud predefinida pero se dispone de información referente a su ubicación.



En el fichero físico la información la guardamos en hojas indexadas, donde cada dato dispone de un localizador. Por ejemplo, en la página 3 línea 3 encontramos el dato correspondiente al año de nacimiento de Ramón González. En el fichero informático la información la guardamos por *bytes*, sabiendo que cada tipo de dato ocupa un número de *bytes* determinado. Así si guardamos *Importe: 3235*, podríamos hacerlo ocupando *11 bytes* desglosados en *9 bytes* (uno por carácter, incluidos los espacios) asociados a "Importe:" como *tipo texto + 2 bytes* (para un tipo *Integer*) asociados al número *3235*.

También podríamos guardarlo todo como tipo texto ocupando *13 bytes*.

Una vez guardada la información, podríamos recuperarla indicándole al ordenador: *Extrae como tipo texto los bytes comprendidos entre 1 y 9 y como número los bytes comprendidos entre 10 y 11.*

Hemos tratado de introducir, a través de la analogía con un fichero físico, enfoques básicos para la comprensión y gestión de ficheros informáticos. La problemática, como se comprenderá, es muy amplia al intervenir numerosos factores (directorios, unidades de disco, sobreescritura de archivos, borrado accidental de datos, etc.) y posibilidades (datos ordenados, parcialmente ordenados o desordenados, diversidad de tipos de datos, tramos de ficheros con registros y campos, etc.).

Nosotros estudiaremos simplificadamente cómo podemos gestionar ficheros con *Visual Basic.*

Próxima entrega: CU00328A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

ACCESO A FICHEROS CON VISUAL BASIC. ACCESO SECUENCIAL, ALEATORIO, BINARIO

Visual Basic nos permite el acceso a ficheros a través de la instrucción *Open*. El acceso a un fichero puede ser:

a) ***Acceso secuencial:*** orientado a su uso en archivos donde la información está dispuesta como tipo texto.

b) ***Acceso aleatorio:*** orientado a su uso en archivos con datos contenidos en registros de longitud fija, que a su vez pueden estar subdivididos en campos. Un campo puede contener un valor numérico o ser tipo texto.

c) ***Acceso binario:*** permite guardar datos con el orden y estructura que se desee, siendo posible acceder a ellos conociendo la posición (número de *byte*) en que se encuentran.

Al existir distintas formas de acceso, se habla de "archivos secuenciales", "archivos aleatorios" y "archivos binarios", aunque esta terminología debe tomarse con precaución. Es decir, un "archivo binario" hace referencia a un archivo donde habitualmente guardamos y extraemos datos usando el acceso binario. Sin embargo, podríamos usar este archivo con otro tipo de acceso. En resumen, un archivo es "información" y la información puede reordenarse, cambiar, manipularse de distintas maneras, etc. Usar un tipo de acceso en un momento dado no obliga a usarlo siempre, aunque así pueda ser.

En la analogía fichero físico - fichero informático, las equivalencias aproximadas serían las de:

- Archivo secuencial = La información no está en campos de longitud predefinida.
- Archivo aleatorio = Existen campos de longitud predefinida, que pueden carecer de información por no estar disponible.
- Archivo binario = La información no está en campos de longitud predefinida pero se dispone de información referente a su ubicación.

Repetimos que todo es lo suficientemente complejo como para que sea difícil dar definiciones exactas. Nosotros vamos a estudiar únicamente una forma de extraer y guardar datos de un fichero secuencial a través de instrucciones simples. El resto de instrucciones y posibilidades de acceso quedará indicado con una lista de términos para que el lector profundice en ellos si más adelante lo estima oportuno.

Próxima entrega: CU00329A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

OPERAR CON FICHEROS

Es frecuente necesitar guardar información en ficheros o recuperar información desde ficheros con Visual Basic. La forma de hacerlo en general es " Abrir -- > Operar -- > Cerrar", pero la sintaxis concreta a utilizar depende de la versión de Visual Basic que estemos empleando:

Para las versiones menos recientes de Visual Basic: describimos a continuación cómo debe procederse.

Para las versiones más recientes de Visual Basic: consulta la entrega CU00329A-2 de este curso (http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61).

Nos vamos a centrar en describir **de forma práctica** como podemos operar con ficheros con el propósito de generar programas que nos permitan ver aplicaciones algorítmicas interesantes. No vamos a explicar todas las posibles formas de operar con ficheros ni todas las instrucciones relacionadas.

OPEN, CLOSE, FREEFILE. INSTRUCCIONES WRITE# Y INPUT#. EJEMPLOS DE USO.

Para manipular información de un fichero, por ejemplo guardar datos en él o leer datos desde él, lo primero que hemos de hacer es abrir el fichero. El proceso a seguir es:

Abrir el fichero → Manipular datos (extraer, guardar, modificar, etc.) → Cerrar el fichero

Open → Manipular datos → Close

A su vez, cada vez que accedemos a un fichero usamos un número como identificador. Podemos considerar que el identificador es una línea de teléfono o canal entre el ordenador y el fichero. Así, para establecer una llamada (abrir un fichero) tenemos que buscar una línea que esté libre. El número de líneas es grande, pero no infinito. Por tanto, podemos abrir muchos archivos al mismo tiempo, pero no todos los que queramos.

La sintaxis que emplearemos será la siguiente:

```
Open [ruta de acceso] For [tipo de acceso] As [número de archivo]
    .
    .
    .
Close
```

Open: es la instrucción que da lugar a la apertura del fichero.

© 2006-2029 aprenderaprogramar.com

Ruta de acceso: es la ruta del archivo que queremos abrir para manipular. Se entiende que dicho archivo se encuentra en una unidad de disco (por ejemplo *C:\problema3.dat* corresponde al fichero *problema3.dat* que se encuentra en la unidad de disco C:). Si el archivo no existe al tratar de abrirlo para escribir datos será creado automáticamente, pero si tratamos de abrirlo para extraer datos aparecerá el mensaje "Error 53: Archivo no encontrado".

For [tipo de acceso]: indica qué acceso utilizamos para manipular el fichero. Vamos a considerar únicamente ficheros secuenciales, con lo que el tipo de acceso podrá ser ***Output*** (para escribir datos en ficheros, eliminando la información previa que pudiera existir), ***Append*** (para escribir datos, conservando la información previa contenida en el fichero) ó ***Input*** (para leer datos contenidos en un fichero).

As [número de archivo]: indicará el "número de línea" que vamos a emplear para la comunicación. Estableceremos un número de línea utilizando la función *FreeFile*, que devuelve un tipo *Integer* que indica el siguiente canal disponible para usar un *Open*.

Close: da lugar al cierre del fichero y a que el número de archivo o línea de comunicación quede libre para una posterior ocasión.

Concretando la escritura nos quedamos con:

a) Escritura de datos en archivo secuencial (no conserva información contenida en archivo).

```
Canal = Freefile
Open "C:\misdatos.dat" For Output As Canal
        .
        .
        .
Close
```

b) Escritura de datos en archivo secuencial (conserva información contenida en archivo).

```
Canal = Freefile
Open "C:\misdatos.dat" For Append As Canal
        .
        .
        .
Close
```

c) Lectura de datos de un archivo secuencial.

```
Canal = Freefile
Open "C:\misdatos.dat" For Input As Canal
        .
        .
        .
Close
```

© 2006-2029 aprenderaprogramar.com

Ya tenemos definido cómo abrimos y cerramos la comunicación con un archivo. Ahora tenemos que ver cómo manipulamos los datos. Para ello nos valemos de las instrucciones *Write#* y *Input#.* Existen distintas variantes para el uso de estas instrucciones. Vamos a considerar únicamente el caso de escritura y extracción de una variable por línea. La sintaxis que emplearemos será:

a) Para escribir en el fichero:

```
Write #[Número de archivo], [Dato a escribir]
```

El dato a escribir puede ser un texto entrecomillado, un valor numérico o una variable numérica o tipo texto.

b) Para extraer datos del fichero:

```
Input #[Número de archivo], [Nombre de variable]
```

El nombre de variable ha de corresponder a una variable tipo texto o tipo numérico que habremos declarado previamente. Es ideal saber qué tipo de dato contiene el fichero para hacer una extracción usando una variable adecuada. Si no dispusiéramos de esta información podríamos usar variables tipo *Variant,* pero no es lo más deseable.

Consideremos un programa a modo de ejemplo (ver el código incluido a continuación al mismo tiempo que se leen las explicaciones):

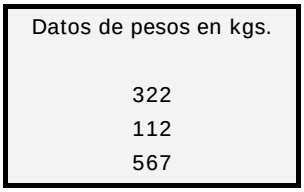
El código comienza con la declaración de las variables que van a intervenir en el programa. Se asignan valores a las variables *Encabezado* (tipo *texto*) y *Dato(1), (2)* y *(3)* (tipo *Integer*). El valor de Canal se establece a través de *FreeFile*, con lo que hay una asignación automática de "línea de comunicación". A continuación se abre el archivo C:\misdatos.dat (en caso de que no existiera previamente el archivo es creado) para escritura (Output). A través de *Write#* se escriben cuatro líneas y se cierra la comunicación.

Se vuelve a abrir comunicación para lectura de datos (*Input*), se asignan los datos a las variables *Textoextraido* y *Datoextraido(1)*, *(2)* y *(3)* y se cierra la comunicación.

© 2006-2029 aprenderaprogramar.com

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Canal%, i%
Dim Encabezado As String
Dim Dato(3) As Integer
Dim Datoextraido(3) As Integer
Dim Textoextraido As String

Private Sub Form_Load()
Show
Encabezado = "Datos de pesos en kgs"
Dato(1) = 322
Dato(2) = 112
Dato(3) = 567

Canal = FreeFile
Open "C:\misdatos.dat" For Output As Canal
Write #Canal, "Datos de pesos en kgs"
Write #Canal, Dato(1)
Write #Canal, Dato(2)
Write #Canal, Dato(3)
Close Canal

Canal = FreeFile
Open "C:\misdatos.dat" For Input As Canal
Input #Canal, Textoextraido
Input #Canal, Datoextraido(1)
Input #Canal, Datoextraido(2)
Input #Canal, Datoextraido(3)
Close Canal

Label1.Alignment = 2
Label1.FontBold = True
Label1 = vbCrLf & Textoextraido & vbCrLf & vbCrLf
For i = 1 To 3
   Label1 = Label1 & Datoextraido(i) & vbCrLf
Next i
```

La información extraída del archivo se muestra en pantalla, donde aparece:



Si abrimos directamente el archivo *C:\misdatos.dat* con un visor como el bloc de notas, el contenido que apreciamos es el siguiente:

© 2006-2029 aprenderaprogramar.com

```
"Datos de pesos en kgs."
322
112
567
```

La escritura mediante *Write#* de valores tipo texto queda delimitada por un entrecomillado que es eliminado en el momento de la extracción.

El tamaño del archivo es de *40 bytes* desglosados en:

- *23 bytes* al texto de la primera línea.
- *2 bytes* al salto de línea y retorno de carro de la primera línea.
- *3 bytes* a la segunda línea.
- *2 bytes* al salto de línea y retorno de carro de la segunda línea.
- *3 bytes* a la tercera línea.
- *2 bytes* al salto de línea y retorno de carro de la tercera línea.
- *3 bytes* a la cuarta línea.
- *2 bytes* al salto de línea y retorno de carro de la cuarta línea.

Se comprueba lo indicado en relación al acceso secuencial: a pesar de guardar variables tipo *Integer*, el espacio ocupado no es el correspondiente a este tipo de variables (*2 bytes*) sino el equivalente a que fueran un texto en el archivo (*1 byte* por carácter).

Existen numerosos términos para el manejo de ficheros que no vamos a abordar.

Algunos términos tienen distinto significado o efecto en función del tipo de acceso a fichero que se utilice o en función de la versión de Visual Basic que se utilice.

La señal de Final de Archivo (EOF) nos resultará útil para extraer datos desde un archivo hasta llegar al final del mismo (punto donde finalizan los datos) la estudiaremos en el apartado correspondiente a "Herramientas de programación con Visual Basic".

Próxima entrega: CU00329A-2 y CU00330A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

OPERAR CON FICHEROS

Es frecuente necesitar guardar información en ficheros o recuperar información desde ficheros con Visual Basic. La forma de hacerlo en general es " Abrir -- > Operar -- > Cerrar", pero la sintaxis concreta a utilizar depende de la versión de Visual Basic que estemos empleando:

Para las versiones menos recientes de Visual Basic: consulta la entrega CU00329A-1 de este curso (http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61)

Para las versiones más recientes de Visual Basic: describimos a continuación cómo debe procederse.

Nos vamos a centrar en describir **de forma práctica** como podemos operar con ficheros con el propósito de generar programas que nos permitan ver aplicaciones algorítmicas interesantes. No vamos a explicar todas las posibles formas de operar con ficheros ni todas las instrucciones relacionadas.

ABRIR UN FICHERO PARA ESCRIBIR EN ÉL

Para manipular información de un fichero, en concreto para guardar datos en él, lo primero que hemos de hacer es abrir el fichero. El proceso a seguir es:

Abrir el fichero --> Manipular datos (borrar lo existente y escribir, o añadir a continuación de lo existente) --> Cerrar el fichero

Open --> Manipular datos --> Close

La sintaxis que emplearemos será la siguiente:

```
Dim nombreInterno As New System.IO.StreamWriter("RutaDelFichero", TrueoFalse)
.
.
.
nombreInterno.Close()
```

nombreInterno: es el nombre que queramos darle a la representación interna que va a usar el programa del fichero. No es el nombre del fichero propiamente dicho, sino un nombre elegido por nosotros para referirnos al fichero.

System.IO.StreamWriter: es el tipo de objeto que el programa va a usar internamente para guardar la información asociada a un fichero que se abre para escritura (write).

RutaDelFichero: indica la ruta en que se localiza el fichero. Por ejemplo "C:\Users\Lenovo\Desktop\misdatosnet.dat" es una ruta. Si no especificamos una ruta completa sino una ruta como "misdatosnet.dat" el fichero se guardará en el directorio por defecto que esté empleando Visual Basic.

© 2006-2029 aprenderaprogramar.com

TrueoFalse: valor booleano donde especificamos si queremos tener activada (True) la identificación del juego de caracteres con que está codificado el archivo, o no tenerla activada (False). Nosotros mantendremos este parámetro como False.

Dim NombreInterno As New…: es la sintaxis con la que creamos el objeto que representa a nuestro fichero.

Concretando la escritura damos a continuación ejemplos prácticos:

a) Escritura de datos en archivo secuencial (creará el archivo si no existe).

```
Dim myFileToWrite As New System.IO.StreamWriter("C:\Users\Asus\Desktop\misdatosnet.dat", False)

        .
        .
        .
myFileToWrite.Close()
```

ABRIR UN FICHERO PARA LEER INFORMACIÓN DESDE ÉL

Para extraer información de un fichero, en concreto para recuperar datos desde él, lo primero que hemos de hacer es abrir el fichero. El proceso a seguir es:

Abrir el fichero --> Manipular datos (extraer información existente) --> Cerrar el fichero

Open --> Manipular datos --> Close

La sintaxis que emplearemos será la siguiente:

```
Dim nombreInterno As New System.IO.StreamReader("RutaDelFichero", TrueoFalse)
.
.
.
nombreInterno.Close()
```

nombreInterno: es el nombre que queramos darle a la representación interna que va a usar el programa del fichero. No es el nombre del fichero propiamente dicho, sino un nombre elegido por nosotros para referirnos al fichero.

System.IO.StreamReader: es el tipo de objeto que el programa va a usar internamente para guardar la información asociada a un fichero que se abre para extracción de datos (lectura, read).

RutaDelFichero: indica la ruta en que se localiza el fichero. Por ejemplo "C:\Users\Lenovo\Desktop\misdatosnet.dat" es una ruta. Si no especificamos una ruta completa sino una ruta como "misdatosnet.dat" el fichero se guardará en el directorio por defecto que esté empleando Visual Basic.

© 2006-2029 aprenderaprogramar.com

TrueoFalse: valor booleano donde especificamos si queremos tener activada (True) la identificación del juego de caracteres con que está codificado el archivo, o no tenerla activada (False). Nosotros mantendremos este parámetro como False.

Dim NombreInterno As New…: es la sintaxis con la que creamos el objeto que representa a nuestro fichero.

Concretando la escritura damos a continuación ejemplos prácticos:

Lectura de datos en archivo secuencial (habrá un error si el archivo no existe).

```
Dim myFileToWrite As New System.IO.StreamWriter("C:\Users\Asus\Desktop\misdatosnet.dat", False)
            .
            .
            .
myFileToWrite.Close()
```

Ya tenemos definido cómo abrimos y cerramos la comunicación con un archivo. Ahora tenemos que ver cómo manipulamos los datos.

ESCRITURA EN FICHEROS

Para escribir en ficheros disponemos de varias instrucciones o posibilidades. Vamos a explicar una forma sencilla de escribir sobre ficheros, basadas en los métodos Write para escribir datos, o WriteLine, para escribir datos línea a línea.

La sintaxis que emplearemos para escribir línea a línea en el fichero será:

```
nombreInterno.WriteLine (datoAEscribir)
```

Donde **nombreInterno** es el nombre interno que estamos usando en nuestro programa como representación simbólica del fichero y **datoAEscribir** es aquello que queremos escribir en el fichero (podrá ser un texto entrecomillado, o una variable alfanumérica o numérica. En el caso de una variable, en el fichero se escribirá el contenido de la variable).

La sintaxis que emplearemos para escribir sin diferenciar líneas será:

```
nombreInterno.Write (datoAEscribir)
```

© 2006-2029 aprenderaprogramar.com

EXTRAER DATOS DE FICHEROS

Para extraer información desde ficheros disponemos de varias instrucciones o posibilidades. Vamos a explicar una forma sencilla de escribir sobre ficheros, basándonos en los métodos Read para leer datos carácter a carácter, o ReadLine, para extraer el contenido de líneas completas.

La sintaxis que emplearemos para extraer línea a línea en el fichero será:

```
variableDondeExtraemos = nombreInterno.ReadLine ()
```

Donde **variableDondeExtraemos** es el nombre de la variable donde va a quedar almacenada la información contenida en la línea del fichero. Si sabemos el tipo de información contenida en el fichero podemos extraer al tipo de variable adecuada. Por ejemplo, un número entero podremos extraerlo a una variable Integer, o un texto a una variable tipo String. **nombreInterno** es el nombre interno que estamos usando en nuestro programa como representación simbólica del fichero. No siempre habrá que extraer el contenido a una variable. Por ejemplo podríamos escribir MsgBox ("Extraido " & nombreInterno.ReadLine(), o usarlo de diferentes maneras sin guardar la información en una variable. No obstante, hemos preferido escribirlo de esta manera porque es el uso más habitual.

La sintaxis que emplearemos para extraer carácter a carácter será:

```
variableDondeExtraemos = variableDondeExtraemos & chr(nombreInterno.Read ())
```

El método Read nos devuelve el valor entero que representa el carácter extraído desde el fichero. Por ejemplo a la letra A le corresponde el número 65. Para obtener el carácter A desde el número correspondiente escribimos chr(65). En este caso la sintaxis que hemos indicado es para añadir a una variable tipo String un carácter extraido desde el fichero. Este no es la única forma posible de utilizar el método Read(), es sólo un ejemplo. Normalmente la extracción carácter a carácter irá combinada con el uso de bucles.

Consideremos un programa a modo de ejemplo (ver el código indicado a continuación al mismo tiempo que se lee la explicación):

El código comienza con la declaración de las variables que van a intervenir en el programa. Se asignan valores a las variables Encabezado (tipo texto) y Dato(1), (2) y (3) (tipo Integer). A continuación se abre el archivo en la ruta especificada (en caso de que no existiera previamente el archivo es creado) para escritura (usamos un StreamWriter). A través del método WriteLine se escriben cuatro líneas y se cierra la comunicación con el método Close().

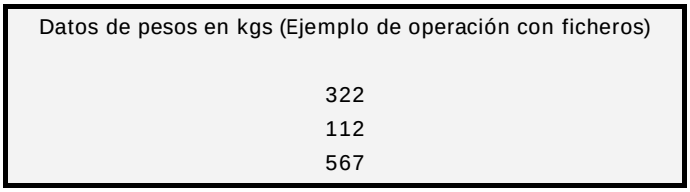
Se vuelve a abrir comunicación para lectura de datos (usamos un StreamReader), se asignan los datos a las variables Textoextraido y Datoextraido(1), (2) y (3) y se cierra la comunicación.

Ejecuta este programa (haz las adaptaciones que sean necesarias en lo relativo a la ruta en tu computador o a la versión de Visual Basic que estés empleando) y comprueba que se crea el archivo con el contenido esperado.

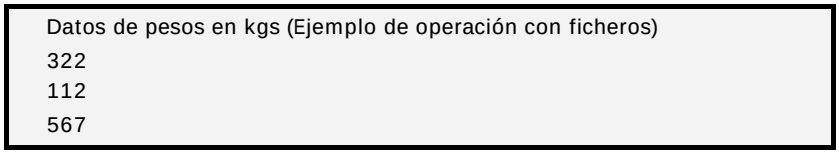
© 2006-2029 aprenderaprogramar.com

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Dim Encabezado As String
    Dim Dato(3) As Integer
    Dim Datoextraido(3) As Integer
    Dim Textoextraido As String

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load

        Dim myFileToWrite As New
System.IO.StreamWriter("C:\Users\Toshiba\Desktop\misdatosnet.dat", False)
        Encabezado = "Datos de pesos en kgs"
        Dato(1) = 322
        Dato(2) = 112
        Dato(3) = 567

        myFileToWrite.WriteLine(Encabezado & " (Ejemplo de operación con ficheros)")
        myFileToWrite.WriteLine(Dato(1))
        myFileToWrite.WriteLine(Dato(2))
        myFileToWrite.WriteLine(Dato(3))
        myFileToWrite.Close()

        Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Toshiba\Desktop\misdatosnet.dat", False)
        Textoextraido = myFileToRead.ReadLine()
        Datoextraido(1) = myFileToRead.ReadLine()
        Datoextraido(2) = myFileToRead.ReadLine()
        Datoextraido(3) = myFileToRead.ReadLine()
        myFileToRead.Close()

        Label1.Font = New Font("Arial", 12, FontStyle.Bold)
        Label1.TextAlign = ContentAlignment.MiddleCenter
        Label1.Text = vbCrLf & Textoextraido & vbCrLf & vbCrLf
        For i = 1 To 3
            Label1.Text = Label1.Text & Datoextraido(i) & vbCrLf
        Next i
    End Sub
End Class
```

La información extraída del archivo se muestra en pantalla, donde aparece:



Si abrimos directamente el archivo *misdatosnet.dat* con un visor como el bloc de notas, el contenido que apreciamos es el siguiente:



© 2006-2029 aprenderaprogramar.com

El tamaño del archivo es de *75 bytes* desglosados de forma aproximada en:

- *58 bytes* al texto de la primera línea.
- *2 bytes* al salto de línea y retorno de carro de la primera línea.
- *3 bytes* a la segunda línea.
- *2 bytes* al salto de línea y retorno de carro de la segunda línea.
- *3 bytes* a la tercera línea.
- *2 bytes* al salto de línea y retorno de carro de la tercera línea.
- *3 bytes* a la cuarta línea.
- *2 bytes* al salto de línea y retorno de carro de la cuarta línea.

Se comprueba lo indicado en relación al acceso secuencial: a pesar de guardar variables tipo *Integer*, el espacio ocupado no es el correspondiente a este tipo de variables (*2 bytes*) sino el equivalente a que fueran un texto en el archivo (*1 byte* por carácter).

Existen numerosos términos para el manejo de ficheros que no vamos a abordar.

Algunos términos tienen distinto significado o efecto en función del tipo de acceso a fichero que se utilice o en función de la versión de Visual Basic que se utilice.

La señal de Final de Archivo (EOF) nos resultará útil para extraer datos desde un archivo hasta llegar al final del mismo (punto donde finalizan los datos) la estudiaremos en el apartado correspondiente a "Herramientas de programación con Visual Basic".

Próxima entrega: CU00330A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

EJERCICIO Nº 1 EJEMPLO LEER FICHERO EN VISUAL BASIC Y HACER CÁLCULOS SENCILLOS.

Crear el código que dé respuesta al siguiente problema planteado: Un operario de una fábrica recibe cada cierto tiempo un depósito cilíndrico de dimensiones variables, que debe llenar de aceite a través de una toma con cierto caudal disponible. Se desea crear un programa que le indique cuánto tiempo transcurrirá hasta el llenado del depósito. El caudal disponible se considera estable para los tiempos que tardan los llenados de depósitos y se encuentra almacenado en un archivo del ordenador, figurando el dato en litros por minuto.

Nota: considerar que el caudal disponible se encuentra en el archivo caudal.dat. Suponer que el archivo caudal.dat se encuentra en la unidad de disco C: (o en aquella ruta que se prefiera), y que el valor de caudal disponible en litros por minuto que proporciona es 112.

SOLUCIÓN

Antes de ejecutar el programa habremos creado el archivo caudal.dat que contendrá un número que representa el valor de caudal.

El código a plantear será diferente según la versión de Visual Basic que estemos utilizando:

a) Para versiones menos recientes de Visual Basic: usaremos la gestión de ficheros basada en las instrucciones Open, Write, Freefile, Write#, Input#, etc.

b) Para las versiones más recientes de Visual Basic usaremos la gestión de ficheros basada en StreamReader, StreamWriter, Write, Read, WriteLine y ReadLine.

Nosotros vamos a exponer el código necesario tanto para versiones menos recientes como para versiones más recientes de Visual Basic (aunque dependiendo de la versión de Visual Basic que estés utilizando es posible que necesites hacer pequeñas adaptaciones).

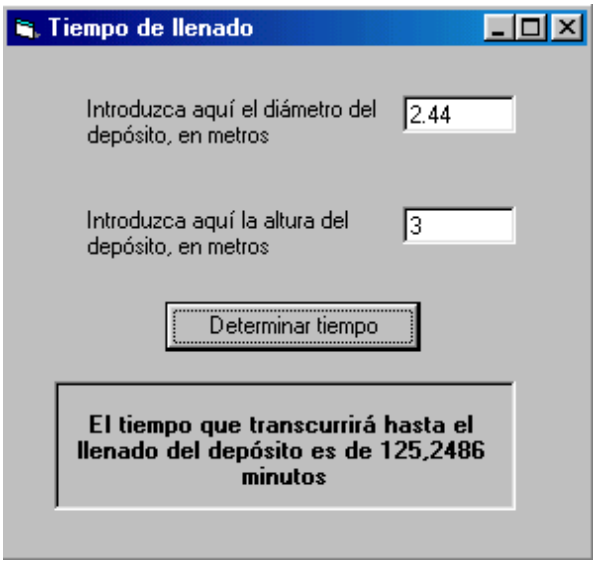
Comenzaremos creando un nuevo proyecto al que añadiremos tres labels (los dos primeros para indicar un mensaje de petición de datos de diámetro y altura y el tercero para mostrar resultados), dos textbox para que el usuario introduzca los datos de diámetro y altura, y finalmente un botón que el usuario deberá pulsar una vez introducidos los datos para obtener los resultados.

© 2006-2029 aprenderaprogramar.com

Código para versiones menos recientes de Visual Basic:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Const Pi As Single = 3.141592654
Dim D!, R!, H!, V!, Vlitros!, tminutos As Single
Dim Canal%
Dim Q As Single

Private Sub Form_Load()
Form1.Caption = "Tiempo de llenado"
CommandCalcular.Caption = "Determinar tiempo"
Label1 = "Introduzca aquí el diámetro del depósito, en metros"
Label2 = "Introduzca aquí la altura del depósito, en metros"
'[Extracción de dato de caudal del archivo]
Canal = FreeFile
Open "C:\caudal.dat" For Input As Canal
Input #Canal, Q
Close Canal
End Sub
Private Sub CommandCalcular_Click()
D = Val(Text1)
H = Val(Text2)
R = D / 2
V = Pi * (R ^ 2) * H
Vlitros = V * 1000
tminutos = Vlitros / Q
Label3.Alignment = 2
Label3.FontBold = True
Label3 = vbCrLf & "El tiempo que transcurrirá hasta el" & _
" llenado del depósito es de " & tminutos & " minutos"
End Sub
```

Gráficamente el resultado será similar a este:

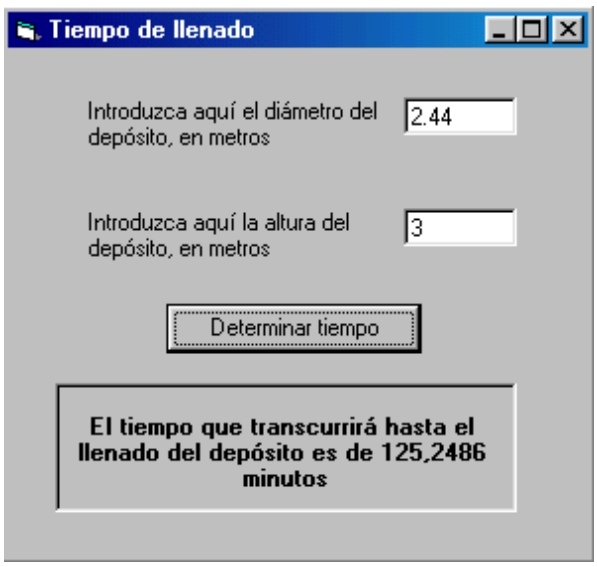


© 2006-2029 aprenderaprogramar.com

Código para versiones más recientes de Visual Basic:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Const Pi As Single = 3.141592654
    Dim D, R, H, V, Vlitros, tminutos As Single
    Dim Q As Single

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
   Me.Text = "Tiempo de llenado"
   CommandCalcular.Text = "Determinar tiempo"
   Label1.Text = "Introduzca aquí el diámetro del depósito, en metros"
   Label2.Text = "Introduzca aquí la altura del depósito, en metros"
   '[Extracción de dato de caudal del archivo]
   Dim myFileToRead As New System.IO.StreamReader("C:\Users\Toshiba\Desktop\caudal.dat", False)
   Q = myFileToRead.ReadLine()
   myFileToRead.Close()
   End Sub

    Private Sub CommandCalcular_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CommandCalcular.Click
        D = Val(TextBox1.Text)
        H = Val(TextBox2.Text)
        R = D / 2
        V = Pi * (R ^ 2) * H
        Vlitros = V * 1000
        tminutos = Vlitros / Q
        Label3.Font = New Font("Arial", 10, FontStyle.Bold)
        Label3.TextAlign = ContentAlignment.MiddleCenter
        Label3.Text = vbCrLf & "El tiempo que transcurrirá hasta el" & _
        " llenado del depósito es de " & tminutos & " minutos"
    End Sub
End Class
```

Gráficamente el resultado será similar a este:



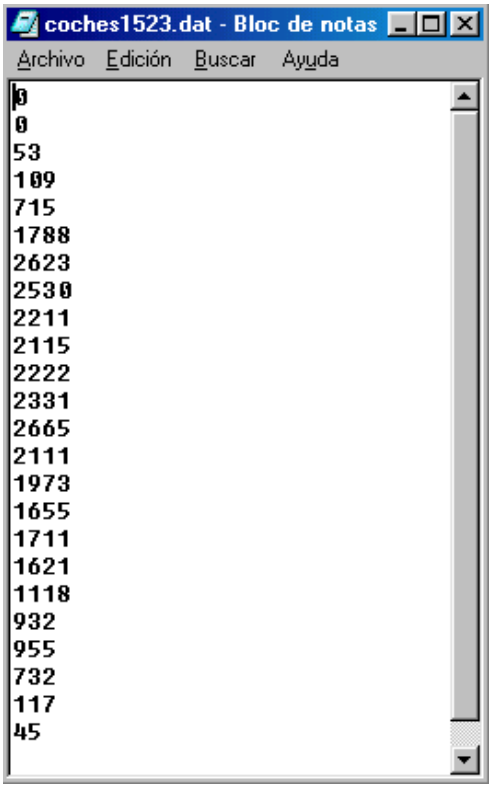
© 2006-2029 aprenderaprogramar.com

EJERCICIO Nº 2 EJEMPLO CON LECTURA DE DATOS DE FICHERO Y USO DE BUCLES FOR Y CONDICIONALES IF THEN

Crear el código correspondiente a este problema. Se ha procedido a realizar un conteo de los coches que han pasado por una vía durante 24 horas. Los resultados se encuentran almacenados en un archivo, en un array definido como cocheshora(1), cocheshora(2)... hasta cocheshora(24) indicando el localizador la hora del día y el contenido de la variable la cantidad de coches que pasaron a esa hora. Se desea desarrollar el pseudocódigo y el diagrama de flujo para un programa que determine:

a) El total de coches que pasaron durante el día.
b) Cuántas horas de tránsito cero hubo y cuáles fueron.
c) Cuál fue la hora de máximo tránsito y qué número de coches pasaron.
d) Cuál fue la última hora a la que pasó un coche y qué número de coches pasaron.

Para simular los datos contenidos en el archivo desde el que ha de leer el programa, crearemos un archivo (usando Visual Basic o manualmente) en la ruta C:\coches1523.dat, ó en aquella ruta que deseemos, cuyo contenido será el siguiente:



© 2006-2029 aprenderaprogramar.com

SOLUCIÓN AL EJERCICIO

Código para versiones menos recientes de Visual Basic:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Canal%, i As Integer
Dim cocheshora(24) As Integer
Dim Cochestotal%, Transitocero%, Maxtransito%, Horamaxtran As Integer
Dim Ultimahora%, Cochesultima As Integer

Private Sub Form_Load()
Form1.Caption = "Coches"
CommandCalcular.Caption = "Leer archivo"
End Sub

Private Sub CommandCalcular_Click()
LabelResultados.Alignment = 2
LabelResultados.FontBold = True
'[Apertura del archivo]
Canal = FreeFile
Open "C:\coches1523.dat" For Input As Canal
For i = 1 To 24
   '[Lectura de datos desde archivo]
   Input #Canal, cocheshora(i)
   Cochestotal = Cochestotal + cocheshora(i)
   If cocheshora(i) = 0 Then
      Transitocero = Transitocero + 1
      LabelResultados = LabelResultados & "La hora " & i & " hubo tránsito cero" & vbCrLf & vbCrLf
   End If
   If cocheshora(i) > Maxtransito Then
      Maxtransito = cocheshora(i)
      Horamaxtran = i
   End If
   If cocheshora(i) > 0 Then
      Ultimahora = i
      Cochesultima = cocheshora(i)
   End If
Next i
Close Canal '[Cerramos el archivo]

LabelResultados = LabelResultados & "El total de coches que" _
& " pasaron durante el día fueron " & Cochestotal & vbCrLf
LabelResultados = LabelResultados & vbCrLf & "Las horas de tránsito cero fueron " & _
Transitocero & vbCrLf
LabelResultados = LabelResultados & vbCrLf & "La hora de máximo tránsito fue la " & _
Horamaxtran & " registrándose el paso de " & Maxtransito & " coches" & vbCrLf
LabelResultados = LabelResultados & vbCrLf & "La última hora a la que pasó un coche" & _
" fue la " & Ultimahora & " registrándose el paso de " & Cochesultima & " coches"
End Sub
```

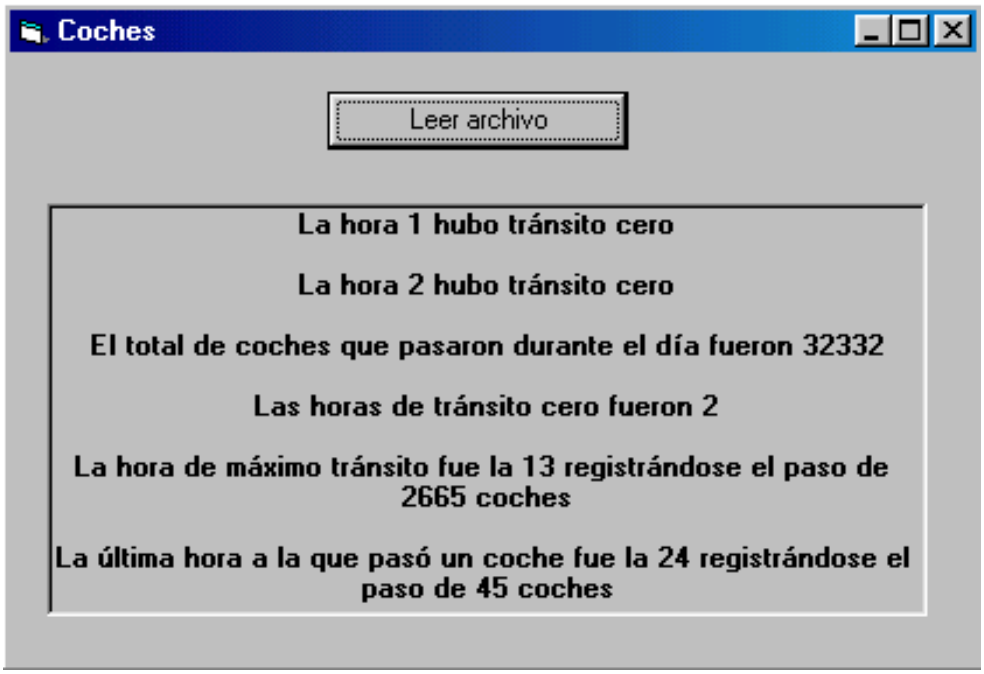
© 2006-2029 aprenderaprogramar.com

Código para versiones más recientes de Visual Basic:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Dim cocheshora(24) As Integer
    Dim Cochestotal, Transitocero, Maxtransito, Horamaxtran As Integer
    Dim Ultimahora, Cochesultima As Integer

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.Text = "Coches"
        CommandCalcular.Text = "Leer archivo"
    End Sub

    Private Sub CommandCalcular_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CommandCalcular.Click

    Dim contador As Integer

    LabelResultados.Font = New Font("Arial", 10, FontStyle.Bold)
    LabelResultados.TextAlign = ContentAlignment.MiddleCenter

   '[Apertura del archivo]
   Dim myFileToRead As New  System.IO.StreamReader("C:\Users\Toshiba\Desktop\coches1523.dat",
False)

   For contador = 1 To 24
       '[Lectura de datos desde archivo]
       cocheshora(contador) = myFileToRead.ReadLine()
       Cochestotal = Cochestotal + cocheshora(contador)
       If cocheshora(contador) = 0 Then
           Transitocero = Transitocero + 1
           LabelResultados.Text = LabelResultados.Text & "La hora " & contador & " hubo
tránsito cero" & vbCrLf & vbCrLf
       End If
       If cocheshora(contador) > Maxtransito Then
           Maxtransito = cocheshora(contador)
           Horamaxtran = contador
       End If
       If cocheshora(contador) > 0 Then
           Ultimahora = contador
           Cochesultima = cocheshora(contador)
       End If
   Next contador

   myFileToRead.Close()

LabelResultados.Text = LabelResultados.Text & "El total de coches que" _
& " pasaron durante el día fueron " & Cochestotal & vbCrLf
LabelResultados.Text = LabelResultados.Text & vbCrLf & "Las horas de tránsito cero fueron " & _
Transitocero & vbCrLf

LabelResultados.Text = LabelResultados.Text & vbCrLf & "La hora de máximo tránsito fue la " & _
Horamaxtran & " registrándose el paso de " & Maxtransito & " coches" & vbCrLf

LabelResultados.Text = LabelResultados.Text &vbCrLf&"La última hora a la que pasó un coche" & _
" fue la " & Ultimahora & " registrándose el paso de " & Cochesultima & " coches"

    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Comentarios: En relación a la lógica del programa, es preferible realizar la apertura y cierre del archivo una sola vez, fuera del bucle, a proceder a abrir y cerrar el archivo 24 veces disponiendo el código de apertura y cierre dentro del bucle. El resultado gráfico sería del tipo:



Próxima entrega: CU00331A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

El objetivo de estos ejercicios es aplicar conocimientos sobre "Herramientas básicas para la programación" a un lenguaje como es Visual Basic. Para su realización debemos tener conocimientos básicos sobre algoritmia, uso de bucles o estructuras iterativas y apertura y cierre de archivos. En algunos casos, propondremos la solución a un ejercicio con distintas estructuras de programación.

EJERCICIO EJEMPLO Nº 1

En un fichero se encuentran las notas de 35 alumnos de una clase en un array Nota(1), Nota(2), Nota(3), ..., Nota(35), establecidas entre cero y 10. Desarrollar el código para un programa que determine:

- El porcentaje de aprobados.
- El porcentaje de suspensos.
- El número de alumnos que han obtenido un 10.
- El número de alumnos que han obtenido un 0.

Considerar que los datos se encuentran en el fichero notas.dat, cuyo contenido es el siguiente: 6 – 7 – 8 – 5 – 6 – 4 – 7 – 8 – 7 – 10 – 0 – 5 – 6 – 7 – 6 – 5 – 0 – 7 – 8 – 10 – 2 – 3 – 4 – 5 – 6 – 7 – 6 – 5 – 4 – 10 – 4 – 3 – 2 – 3 – 4 (suponemos que cada dato está en una línea, aquí los ponemos separados por guiones para no ocupar 35 líneas).

SOLUCIÓN

El código a plantear será diferente según la versión de Visual Basic que estemos utilizando:

a) Para versiones menos recientes de Visual Basic: usaremos la gestión de ficheros basada en las instrucciones Open, Write, Freefile, Write#, Input#, etc.

b) Para las versiones más recientes de Visual Basic usaremos la gestión de ficheros basada en StreamReader, StreamWriter, Write, Read, WriteLine y ReadLine.

Nosotros vamos a exponer el código necesario tanto para versiones menos recientes como para versiones más recientes de Visual Basic (aunque dependiendo de la versión de Visual Basic que estés utilizando es posible que necesites hacer pequeñas adaptaciones).

Comenzaremos creando un nuevo proyecto al que añadiremos un label para mostrar los resultados y un botón que el usuario deberá pulsar para comenzar a leer los datos desde el archivo y procesarlos.

© 2006-2029 aprenderaprogramar.com

Código para versiones menos recientes de Visual Basic:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Canal As Integer
Dim i As Integer
Dim Nota(35) As Single
Dim Aprobados!, Suspensos As Single
Dim Dieces%, Ceros As Integer

Private Sub Form_Load()
Form1.Caption = "Notas"
CommandCalcular.Caption = "Leer archivo"
End Sub

Private Sub CommandCalcular_Click()
LabelResultados.Alignment = 2
LabelResultados.FontBold = True
'[Apertura del archivo]
Canal = FreeFile
Open "C:\notas.dat" For Input As Canal
For i = 1 To 35
   '[Lectura de datos desde archivo]
   Input #Canal, Nota(i)
   If Nota(i) >= 5 Then
      Aprobados = Aprobados + 1
   Else
      Suspensos = Suspensos + 1
   End If
   If Nota(i) = 10 Then
      Dieces = Dieces + 1
   End If
   If Nota(i) = 0 Then
      Ceros = Ceros + 1
   End If
Next i
Close Canal '[Cerramos el archivo]

LabelResultados = vbCrLf & "% de aprobados es " & (Aprobados / 35) * 100 & vbCrLf
LabelResultados = LabelResultados & vbCrLf & "% de suspensos es " & (Suspensos / 35) * 100 _
& vbCrLf
LabelResultados = LabelResultados & vbCrLf & "Obtienen 10 un total de " & Dieces & _
" alumnos" & vbCrLf
LabelResultados = LabelResultados & vbCrLf & "Obtienen 0 un total de " & Ceros & " alumnos"
End Sub
```

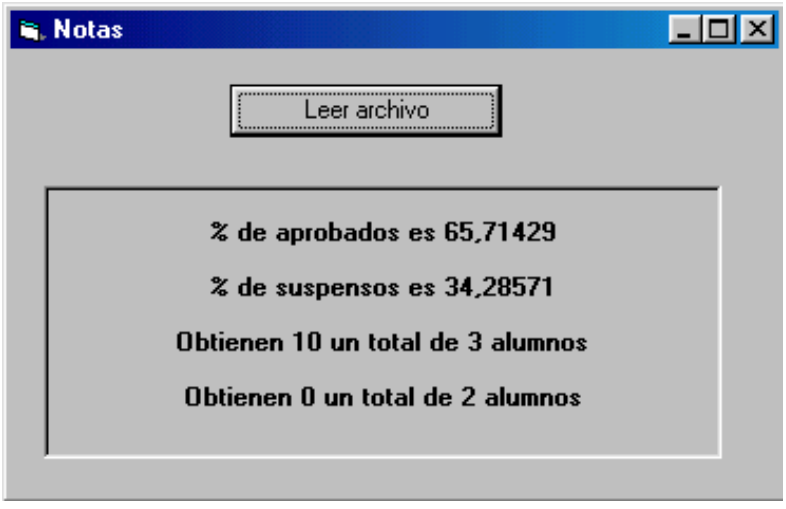
El resultado gráfico es el que se muestra más abajo.

© 2006-2029 aprenderaprogramar.com

Código para versiones más recientes de Visual Basic:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.Text = "Notas"
        Button1.Text = "Leer archivo"
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
   Dim contador As Integer
   Dim Nota(35) As Single
   Dim Aprobados, Suspensos As Single
   Dim Dieces, Ceros As Integer

   LabelResultados.Font = New Font("Arial", 10, FontStyle.Bold)
   LabelResultados.TextAlign = ContentAlignment.MiddleCenter
   '[Apertura del archivo, ojo: poner la ruta que sea correcta]
   Dim myFileToRead As New System.IO.StreamReader("C:\Users\Toshiba\Desktop\notas.dat", False)
   For contador = 1 To 35
            '[Lectura de datos desde archivo]
            Nota(contador) = myFileToRead.ReadLine()
            If Nota(contador) >= 5 Then
                Aprobados = Aprobados + 1
            Else
                Suspensos = Suspensos + 1
            End If
            If Nota(contador) = 10 Then
                Dieces = Dieces + 1
            End If
            If Nota(contador) = 0 Then
                Ceros = Ceros + 1
            End If
   Next contador
   myFileToRead.Close()

   LabelResultados.Text = vbCrLf & "% de aprobados es " & (Aprobados / 35) * 100 & vbCrLf
   LabelResultados.Text = LabelResultados.Text & vbCrLf & "% de suspensos es " & (Suspensos /
35) * 100 & vbCrLf
 LabelResultados.Text = LabelResultados.Text & vbCrLf & "Obtienen 10 un total de " & Dieces &
 " alumnos" & vbCrLf
    LabelResultados.Text = LabelResultados.Text & vbCrLf & "Obtienen 0 un total de " & Ceros &
" alumnos"
    End Sub
End Class
```

Y el resultado gráfico:



© 2006-2029 aprenderaprogramar.com

El bucle se puede plantear igualmente con un **Do While**. Este sería el código:

Para versiones menos recientes de VB:

```
'[Curso VB aprenderaprogramar.com]
i = 1
'[Apertura del archivo]
Canal = FreeFile
Open "C:\notas.dat" For Input As Canal
Do While i <= 35
   '[Lectura de datos desde archivo]
   Input #Canal, Nota(i)
   If Nota(i) >= 5 Then
      Aprobados = Aprobados + 1
   Else
      Suspensos = Suspensos + 1
   End If
   If Nota(i) = 10 Then
      Dieces = Dieces + 1
   End If
   If Nota(i) = 0 Then
      Ceros = Ceros + 1
   End If
   i = i + 1
Loop
Close Canal '[Cerramos el archivo]
```

Para versiones más recientes de VB:

```
'[Curso VB aprenderaprogramar.com]
 contador = 1
'[Apertura del archivo, ojo: poner ruta que sea
correcta]
Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Toshiba\Desktop\not
as.dat", False)
        Do While contador <= 35
            '[Lectura de datos desde archivo]
            Nota(contador) = myFileToRead.ReadLine()
            If Nota(contador) >= 5 Then
                Aprobados = Aprobados + 1
            Else
                Suspensos = Suspensos + 1
            End If
            If Nota(contador) = 10 Then
                Dieces = Dieces + 1
            End If
            If Nota(contador) = 0 Then
                Ceros = Ceros + 1
            End If
            contador = contador + 1
        Loop
myFileToRead.Close()
```

EJERCICIO EJEMPLO Nº 2

En un fichero tenemos las notas de 35 alumnos de una clase en un array Nota(1), Nota(2), ..., Nota(35), establecidas entre cero y 10. Se desea desarrollar un programa que determine la nota media.

Considerar que los datos se encuentran en el fichero notas.dat, cuyo contenido es el siguiente: 6 – 7 – 8 – 5 – 6 – 4 – 7 – 8 – 7 – 10 – 0 – 5 – 6 – 7 – 6 – 5 – 0 – 7 – 8 – 10 – 2 – 3 – 4 – 5 – 6 – 7 – 6 – 5 – 4 – 10 – 4 – 3 – 2 – 3 – 4 (suponemos que cada dato está en una línea, aquí los ponemos separados por guiones para no ocupar 35 líneas).

SOLUCIÓN

Comenzaremos creando un nuevo proyecto al que añadiremos un label para mostrar los resultados y un botón que el usuario deberá pulsar para comenzar a leer los datos desde el archivo y procesarlos.

© 2006-2029 aprenderaprogramar.com

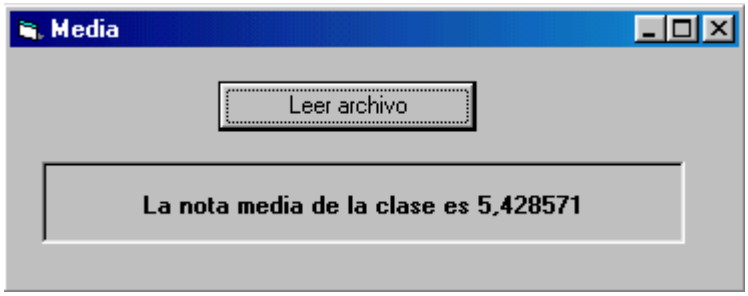
Para versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim Canal%, i As Integer
Dim Nota(35) As Single
Dim Sumatorio!, Media As Single

Private Sub Form_Load()
Form1.Caption = "Media"
CommandCalcular.Caption = "Leer
archivo"
End Sub

Private Sub CommandCalcular_Click()
LabelResultados.Alignment = 2
LabelResultados.FontBold = True
'[Apertura del archivo]
Canal = FreeFile
Open "C:\notas.dat" For Input As Canal
For i = 1 To 35
   '[Lectura de datos desde archivo]
   Input #Canal, Nota(i)
   Sumatorio = Sumatorio + Nota(i)
Next i
Close Canal '[Cerramos el archivo]
Media = Sumatorio / 35
LabelResultados = vbCrLf & "La nota
media de la clase es " & Media
End Sub
```

Para versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Me.Text = "Media"
        Button1.Text = "Leer archivo"
    End Sub

Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Dim contador As Integer
        Dim Nota(35) As Single
        Dim Sumatorio, Media As Single
LabelResultados.Font = New Font("Arial", 10,
FontStyle.Bold)
LabelResultados.TextAlign = ContentAlignment.MiddleCenter
'[Apertura del archivo, ojo: poner ruta correcta]
Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Toshiba\Desktop\notas
.dat", False)
        For contador = 1 To 35
            '[Lectura de datos desde archivo]
            Nota(contador) = myFileToRead.ReadLine()
            Sumatorio = Sumatorio + Nota(contador)
        Next contador
        myFileToRead.Close()
        Media = Sumatorio / 35
LabelResultados.Text = vbCrLf & "La nota media de la
clase es " & Media
    End Sub
End Class
```

Gráficamente:



Próxima entrega: CU00332A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

El objetivo de estos ejercicios es aplicar conocimientos sobre "Herramientas básicas para la programación" a un lenguaje como es Visual Basic. Para su realización debemos tener conocimientos básicos sobre algoritmia, uso de bucles o estructuras iterativas y apertura y cierre de archivos. En algunos casos, propondremos la solución a un ejercicio con distintas estructuras de programación.

EJERCICIO EJEMPLO USO DE FORMS, LABELS, INPUTBOX, MSGBOX

Desarrollar el código que dé respuesta al siguiente problema:

Una central distribuye cal hacia diferentes almacenes sucursales. Disponen de un muelle de carga a donde van llegando sacas de cal de entre 3000 y 9000 Kgs, con pesos variables en función de las circunstancias de la producción. La empresa dispone de una flota de camiones con capacidades de carga de entre 18000 y 28000 Kgs.

Se pretende establecer un protocolo consistente en cargar 20 camiones diarios. Cada camión se quiere cargar como máximo a su límite de capacidad debiendo partir si con la siguiente saca en la línea de producción se fuera a exceder su capacidad.

La empresa quiere desarrollar un programa que le pida al operario encargado de carga la capacidad del camión y el peso de las sacas, indicándole si debe cargar la saca o despachar el camión para comenzar a cargar otro, y que emita un aviso cuando se complete la carga de 20 camiones.

SOLUCIÓN

Código para versiones menos recientes de Visual Basic:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim i%, j%
Dim Capacidad%, Pesosaca%, Cargado As Integer

Private Sub Form_Load()
Show
Form1.Caption = "Distribución de cal"
For i = 1 To 20
   Capacidad = InputBox("Introduzca la capacidad del camión nº " & i & " en kgs", "Dato capacidad")
   Do While(Cargado + Pesosaca) <= Capacidad
      If Pesosaca > 0 Then
         MsgBox("Cargue la saca nº " & j)
         Cargado = Cargado + Pesosaca
      End If
      j = j + 1
      Pesosaca = InputBox("Introduzca el peso de la saca nº " & j & " en kgs", "Dato saca")
   Loop
   If i < 20 Then MsgBox("No cargue todavía. Despache el camión nº " & i & " y ponga" & _
   " a cargar el camión nº " & i + 1)
   Cargado = 0
Next i
Labelfinal.Alignment = 2
Labelfinal.FontBold = True
Labelfinal = "No cargue esta saca. Despache el camión " & i - 1 & vbCrLf & "FINALIZADA" & _
" LA CARGA DEL DÍA"
End Sub
```

© 2006-2029 aprenderaprogramar.com

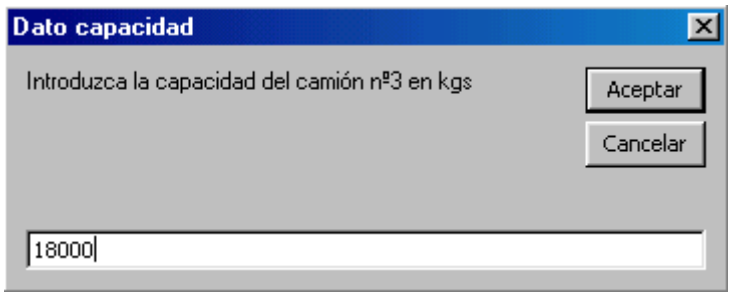
Antes de ejecutar el código habremos incorporado al formulario en la vista de diseño un Label al que hemos denominado Labelfinal. El resultado gráfico se muestra más abajo.

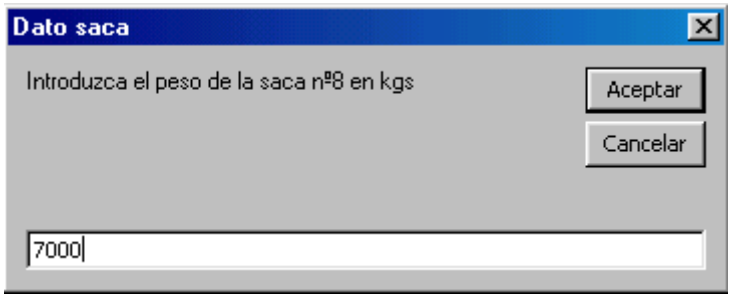
Código para versiones más recientes de Visual Basic:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
   Dim i, j As Integer
   Dim Capacidad, Pesosaca, Cargado As Integer
   Me.Text = "Distribución de cal"
   For i = 1 To 20
   Capacidad = InputBox("Introduzca la capacidad del camión nº " & i & " en kgs", "Dato
capacidad")
   Do While (Cargado + Pesosaca) <= Capacidad
   If Pesosaca > 0 Then
               MsgBox("Cargue la saca nº " & j)
               Cargado = Cargado + Pesosaca
   End If
   j = j + 1
   Pesosaca = InputBox("Introduzca el peso de la saca nº " & j & " en kgs", "Dato saca")
   Loop
   If i < 20 Then MsgBox("No cargue todavía. Despache el camión nº " & i & " y ponga" & _
       " a cargar el camión nº " & i + 1)
   Cargado = 0
   Next i
Labelfinal.Font = New Font("Arial", 10, FontStyle.Bold)
Labelfinal.TextAlign = ContentAlignment.MiddleCenter
Labelfinal.Text = "No cargue esta saca. Despache el camión " & i - 1 &vbCrLf& "FINALIZADA" & _
        " LA CARGA DEL DÍA"
    End Sub

End Class
```

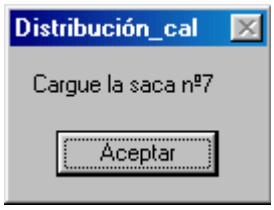
Comentarios: el programa podría mejorarse creando restricciones para que no se puedan introducir pesos de sacas o capacidad de carga de camiones fuera de los rangos permitidos.

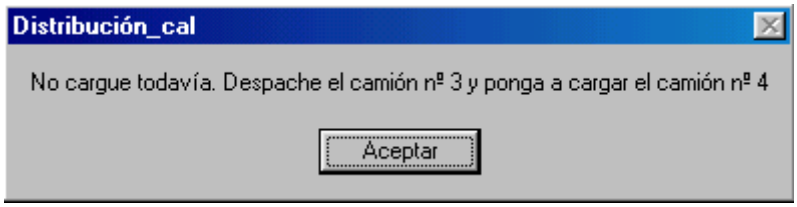
El aspecto gráfico es el siguiente:

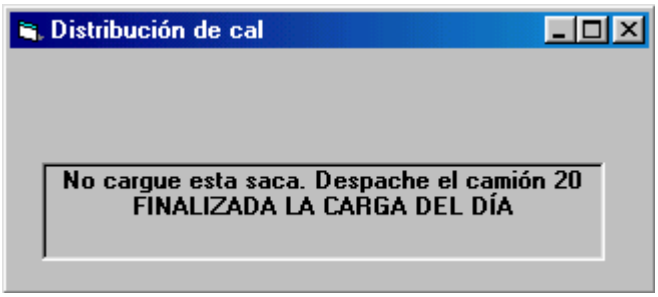




© 2006-2029 aprenderaprogramar.com







EJERCICIO EJEMPLO USO DO WHILE … LOOP, LABELS, INPUTBOX

Desarrollar el código que resuelva el siguiente problema: una ONG tiene puntos de reparto de vacunas que se pretende funcionen de la siguiente manera. Cada día, empezar con 1000 vacunas disponibles y a través de un programa que controla las entregas avisar si el inventario baja de 200 unidades. El programa debe partir de que existen 1000 vacunas e ir pidiendo al usuario las cantidades de vacunas entregadas hasta avisar cuando el inventario baje de 200 unidades.

SOLUCIÓN

Código para versiones menos recientes de Visual Basic:

```
Rem Curso Visual Basic aprenderaprogramar.com [Control de vacunas]

Option Explicit
Dim Existencias%, Entregadas%

Private Sub Form_Load()
Show
Form1.Caption = "Control de vacunas"
Existencias = 1000
Do While Existencias >= 200
   Entregadas = InputBox("Introduzca el número de uds entregadas", "Dato entrega")
   Existencias = Existencias - Entregadas
Loop
Labelfinal.Alignment = 2
Labelfinal.FontBold = True
Labelfinal = vbCrLf & "El inventario ha bajado de 200 uds" & vbCrLf & "Debe comunicarlo"
End Sub
```

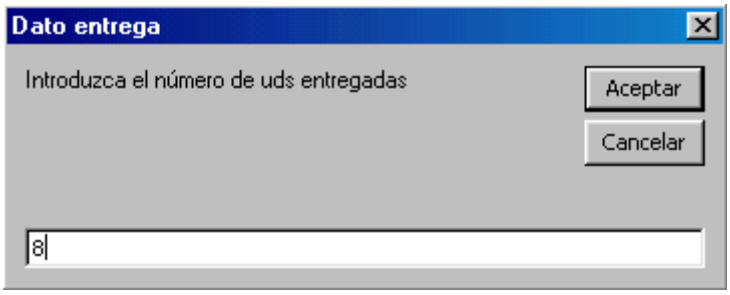
© 2006-2029 aprenderaprogramar.com

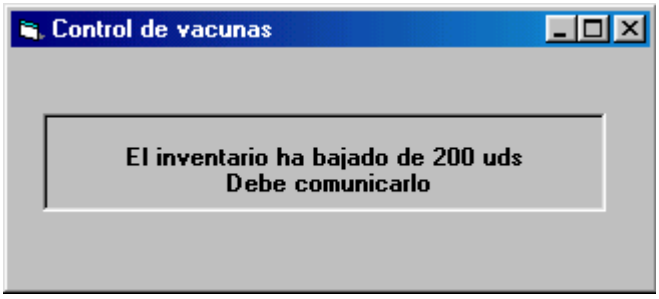
Antes de ejecutar el código habremos incorporado al formulario en la vista de diseño un Label al que hemos denominado Labelfinal. El resultado gráfico se muestra más abajo.

Código para versiones más recientes de Visual Basic:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim Existencias, Entregadas As Integer
        Me.Text = "Control de vacunas"

        Existencias = 1000
        Do While Existencias >= 200
            Entregadas = InputBox("Introduzca el número de uds entregadas", "Dato entrega")
            Existencias = Existencias - Entregadas
        Loop

  Labelfinal.Font = New Font("Arial", 10, FontStyle.Bold)
  Labelfinal.TextAlign = ContentAlignment.MiddleCenter
  Labelfinal.Text = vbCrLf & "El inventario ha bajado de 200 uds" & vbCrLf & "Debe comunicarlo"
  End Sub

End Class
```

El resultado gráfico:





Próxima entrega: CU00333A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

El objetivo de estos ejercicios es aplicar conocimientos sobre "Herramientas básicas para la programación" a un lenguaje como es Visual Basic. Para su realización debemos tener conocimientos básicos sobre algoritmia, uso de bucles o estructuras iterativas y apertura y cierre de archivos.

EJERCICIO EJEMPLO USO DE ARRAYS DE CONTROLES Y FUNCIÓN VAL

Desarrollar el código que dé respuesta al siguiente problema:

Un vivero forestal actualiza cada seis meses los precios de la planta que vende en función de los valores oficiales de inflación mensual. Desean desarrollar un programa que proporcione el precio actualizado a partir del precio anterior y los valores de inflación.

Consideramos que una vez transcurre el mes el nuevo precio es:

$$\text{Precio} = \text{Precio} \times \frac{\text{inflación mes}}{100} + \text{Precio} = \text{Precio} \times \left(1 + \frac{\text{inflación mes}}{100}\right)$$

El siguiente mes el precio a aplicar ya no es el inicial, sino el precio afectado por la inflación.

SOLUCIÓN

Vamos a trabajar con matrices de controles. Código para las versiones menos recientes de Visual Basic:

```
'Curso Visual Basic aprenderaprogramar.com

Option Explicit
Dim i As Integer
Dim Actual!, Nuevo As Single
Dim inf(6) As Single

Private Sub Form_Load()
Show
Form1.Caption = "Precios de planta"
CommandCalcular.Caption = "Calcular precio"
LabelPactual.Caption = "Introduzca aquí el precio actual"
For i = 1 To 6
   Label1(i).Caption = "Introduzca aquí la inflación del mes " & i
Next
End Sub
```

```
Private Sub CommandCalcular_Click()
Nuevo = Val(TextPactual)
For i = 1 To 6
   inf(i) = Val(Text1(i))
   Nuevo = Nuevo * (1 + (inf(i)) / 100)
Next i
Labelfinal.Alignment = 2
Labelfinal.FontBold = True
Labelfinal = vbCrLf & "El precio nuevo del producto debe ser " & Nuevo
End Sub
```

© 2006-2029 aprenderaprogramar.com

El resultado gráfico se muestra más abajo.

En el diseño del formulario deberemos haber creado un Label denominado LabelPactual y un textBox denominado TextPactual y haberlos situado en la parte superior. Además necesitaremos un botón que el usuario debe pulsar una vez introducidos los datos y un label para mostrar los resultados, que hemos denominado Labelfinal y que hemos situado en la parte inferior de la ventana.

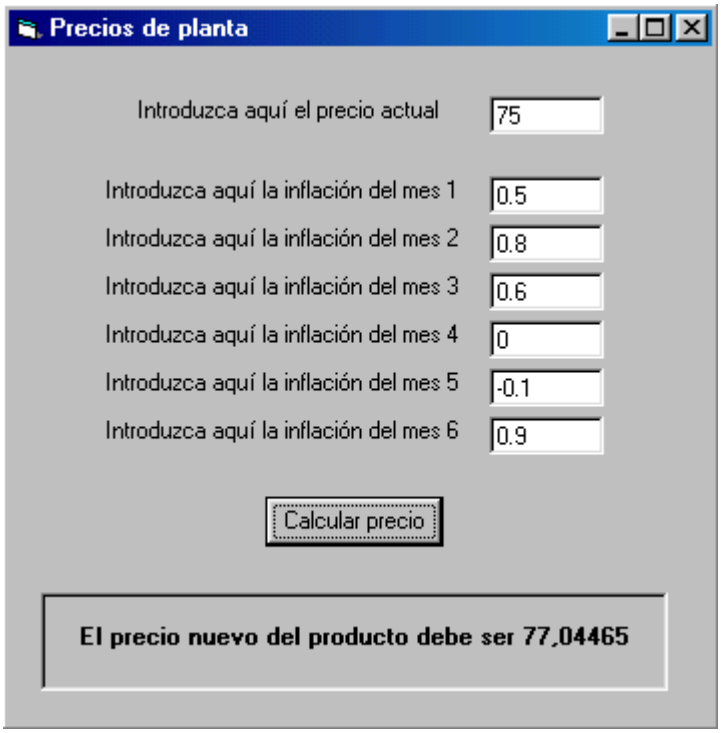
Código para versiones más recientes de Visual Basic:

```
REM Curso Visual Basic aprenderaprogramar.com
Public Class Form1
    Dim LabelArray(6) As Label
    Dim TextBoxArray(6) As TextBox
    Dim Actual, Nuevo As Single
    Dim inf(6) As Single

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim i As Integer
        Me.Text = "Precios de planta"
        ButtonCalcular.Text = "Calcular precio"
        LabelPactual.Text = "Introduzca aquí el precio actual"
        For i = 1 To 6
            LabelArray(i) = New Label
            TextBoxArray(i) = New TextBox
            LabelArray(i).Text = "Introduzca aquí la inflación del mes " & i
            LabelArray(i).Width = 220 : TextBoxArray(i).Width = 100
            LabelArray(i).Height = 30 : TextBoxArray(i).Height = 30
            TextBoxArray(i).AutoSize = False
            LabelArray(i).TextAlign = ContentAlignment.MiddleCenter
            LabelArray(i).BorderStyle = BorderStyle.FixedSingle
            LabelArray(i).Location = New Point(100, i * 50 + 50)
            TextBoxArray(i).Location = New Point(360, i * 50 + 50)
            Me.Controls.Add(LabelArray(i)) : Me.Controls.Add(TextBoxArray(i))
        Next
    End Sub

    Private Sub ButtonCalcular_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonCalcular.Click
        Dim i As Integer
        Nuevo = Val(TextPactual.Text)
        For i = 1 To 6
            inf(i) = Val(TextBoxArray(i).Text)
            Nuevo = Nuevo * (1 + (inf(i)) / 100)
        Next
        Labelfinal.TextAlign = ContentAlignment.MiddleCenter
        Labelfinal.Font = New Font("Arial", 10, FontStyle.Bold)
        Labelfinal.Text = vbCrLf & "El precio nuevo del producto debe ser " & Nuevo
    End Sub
End Class
```

Comentarios: en lugar de arrays de controles podríamos haber usado controles simples como Label1, Label2, Label3, etc. pero en este caso no podríamos haber usado bucles para realizar los cálculos, sino que tendríamos que repetir los cálculos.

© 2006-2029 aprenderaprogramar.com

Gráficamente el resultado será similar a este:



EJERCICIO EJEMPLO USO DE GENERACIÓN DE NÚMEROS ALEATORIOS Y USO DE ARCHIVOS

Desarrollar el código que dé respuesta al siguiente problema:

En un complejo industrial se desea incentivar a los encargados de turnos de producción. A cada encargado le corresponde un turno. Hay 18 turnos, y por cada uno de ellos 40 operarios. Para evaluar a los encargados se ha hecho una encuesta entre los trabajadores. Se les ha pedido que evalúen a su encargado de 0 a 10 siendo el 5 el aprobado. Los resultados están almacenados en un array tipo Nota(turno, operario) donde turno indica el número de turno y operario el número de operario dentro del turno. Desarrollar un programa que determine en qué turnos todos los operarios del mismo han aprobado al encargado.

Nota: las notas las generaremos aleatoriamente en un fichero con un pequeño programa auxiliar.

SOLUCIÓN

Generaremos una simulación del archivo de evaluación. Para ello haremos una creación de datos con base aleatoria pero "dirigida". Sabiendo que cada turno tiene 40 empleados, dejaremos una generación aleatoria entre 0 y 10 para todos los turnos excepto los número 3 y 18 (asociados a los datos 81 a 120 y 681 a 720), en los que los datos generados “obligaremos” a que se encuentren entre 5 y 10, de modo que sabemos de antemano que todos los empleados de los turnos nº3 y nº18 "han aprobado" a su encargado. Los códigos para crear el archivo de evaluación y el programa serían los siguientes:

© 2006-2029 aprenderaprogramar.com

Código versiones menos recientes de VB:

```
'Curso Visual Basic aprenderaprogramar.com
'[Archivo de evaluación]

Option Explicit
Dim Canal As Integer
Dim i As Integer

Private Sub Form_Load()
Show
Canal = FreeFile
Open "C:\evaluación.dat" For Output As Canal
For i = 1 To 720
   Randomize
   If i < 81 Or i > 120 And i < 681 Then
      Write #Canal, Int((11) * Rnd)
   Else
      Write #Canal, Int((10 - 5 + 1) * Rnd + 5)
   End If
Next i
Close
End Sub
```

Código versiones más recientes de Visual Basic:

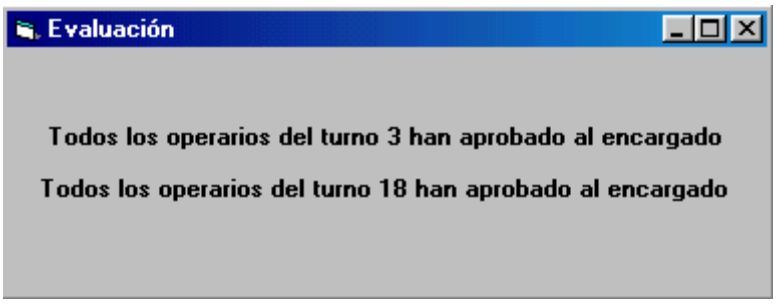
```
REM Curso Visual Basic aprenderaprogramar.com
Public Class Form1
    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim i As Integer
        Button1.Text = "Generar archivo de datos"
    End Sub

Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Dim myFileToWrite As New
System.IO.StreamWriter("C:\Users\Toshiba\Desktop\ev
aluacion.dat", False)

  For i = 1 To 720
      Randomize()
      If i < 81 Or i > 120 And i < 681 Then
          myFileToWrite.WriteLine(Int(11 * Rnd()))
      Else
          myFileToWrite.WriteLine(Int((10 - 5 + 1)
* Rnd() + 5))
      End If
  Next i
  myFileToWrite.Close()
 End Sub
End Class
```

```
'Curso Visual Basic aprenderaprogramar.com
'[Evaluación de encargados]
Option Explicit
Dim Canal As Integer
Dim i%, j As Integer
Dim Nota(18, 40) As Integer
Dim Haysuspensos As Boolean
Private Sub Form_Load()
Show
Labelfinal.Alignment = 2
Labelfinal.FontBold = True
Canal = FreeFile
Open "C:\evaluación.dat" For Input As Canal
For i = 1 To 18
   For j = 1 To 40
      Input #Canal, Nota(i, j)
      If Nota(i, j) < 5 Then
         Haysuspensos = True
      End If
   Next j
   If Haysuspensos = False Then
      Labelfinal = Labelfinal & vbCrLf & "Todos
los operarios del turno " & i & _
      " han aprobado al encargado" & vbCrLf
   End If
   Haysuspensos = False
Next i
Close
End Sub
```

```
REM Curso Visual Basic aprenderaprogramar.com
Public Class Form1
    Dim Nota(18, 40) As Integer
    Dim Haysuspensos As Boolean

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
Labelfinal.TextAlign = ContentAlignment.MiddleCenter
Labelfinal.Font = New Font("Arial", 10, FontStyle.Bold)
Labelfinal.Text = ""
Button1.Text = "Generar estadísticas"
    End Sub

    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Dim i, j As Integer
        Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Toshiba\Desktop\eva
luacion.dat", False)
        For i = 1 To 18
            For j = 1 To 40
                Nota(i, j) = myFileToRead.ReadLine()
                If Nota(i, j) < 5 Then
                    Haysuspensos = True
                End If
            Next j
            If Haysuspensos = False Then
                Labelfinal.Text = Labelfinal.Text &
vbCrLf & "Todos los operarios del turno " & i &
" han aprobado al encargado" & vbCrLf
            End If
            Haysuspensos = False
        Next i
        myFileToRead.Close()
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Gráficamente obtendríamos un mensaje al ejecutar el programa que nos indicaría en qué turnos todos los operarios han aprobado al encargado:



Próxima entrega: CU00334A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

El objetivo de estos ejercicios es aplicar conocimientos sobre "Herramientas básicas para la programación" a un lenguaje como es Visual Basic. Para su realización debemos tener conocimientos básicos sobre algoritmia, uso de bucles o estructuras iterativas y apertura y cierre de archivos.

EJERCICIO RESUELTO Nº 1 EJEMPLO DE USO DE ARRAYS ESTÁTICOS, DO WHILE … LOOP, IF … THEN, USO DE FICHEROS…

Generar el código que permita solucionar el siguiente problema:

Un fabricante de maquinaria envía un archivo a un almacén donde se encuentran los números de serie (5 dígitos) de una partida de maquinaria enviada, organizados en Dato(1), Dato(2), ..., Dato(n). Informa de lo siguiente: el archivo empieza con palas cargadoras siendo la correspondiente al número de serie 78345 la última, sigue con retroexcavadoras siendo el número de serie 33287 el último, y termina con dumpers siendo el número de serie 63287 el último.

Se quiere desarrollar un programa que muestre el tipo de máquina y todos los números de serie correspondientes.

Nota: considerar que la cantidad de datos es igual o inferior a 50, y que se encuentran en el fichero maquinaria.dat, que deberá haber sido creado previamente. Probar un caso supuesto con estos datos:

Dato	Nº de inventario
1	11252
2	22541
3	78345
4	47854
5	33287
6	25478
7	63287

SOLUCIÓN

Cuando no conocemos a priori el número de datos de un array, como es la situación de que tenemos una lista de números de serie delimitados por señaleros (centinelas), sin saber un número exacto, lo más habitual será declarar arrays dinámicos y redimensionarlos a medida que vamos extrayendo paquetes de datos. Esto lo veremos más adelante. En este caso usaremos otra opción: crear un array de rango 50 en el cual usaremos una cantidad de elementos del array igual o inferior al rango. Puede ocurrir que la mayor parte de los elementos del array queden sin ser ocupados. Consideremos un fichero que contiene los siete datos indicados en el enunciado, creado previamente.

© 2006-2029 aprenderaprogramar.com

Necesitaremos un label para mostrar los resultados, al que denominaremos Labelfinal. Consideraremos que el código se ejecuta nada más abrir el programa, sin necesidad de pulsar un botón. También debemos conocer la ruta en que se ubica el archivo de datos y escribirla correctamente.

Código (versiones menos recientes de VB):

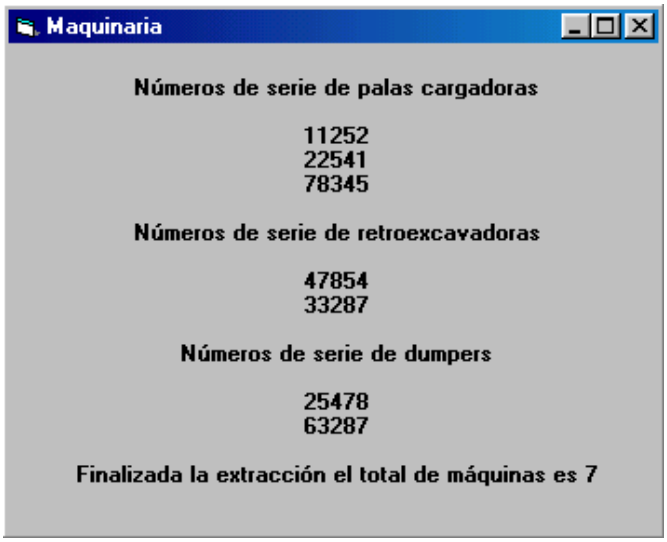
```
'Curso Visual Basic aprenderaprogramar.com
'[Maquinaria]

Option Explicit
Dim Canal As Integer
Dim i As Integer
Dim Dato(50) As Single

Private Sub Form_Load()
Show
Form1.Caption = "Maquinaria"
Labelfinal.Alignment = 2
Labelfinal.FontBold = True
i = 1
Canal = FreeFile
Open "C:\maquinaria.dat" For Input As Canal
Input #Canal, Dato(i)
Labelfinal = "Números de serie de palas
                    cargadoras" & vbCrLf &
                    vbCrLf
Do While Dato(i) <> 63287
   Labelfinal = Labelfinal & Dato(i) & vbCrLf
   If Dato(i) = 78345 Then
      Labelfinal = Labelfinal & vbCrLf &
"Números de serie de retroexcavadoras" &
vbCrLf & vbCrLf
   End If
   If Dato(i) = 33287 Then
      Labelfinal = Labelfinal & vbCrLf &
"Números de serie de dumpers" & vbCrLf &
vbCrLf
   End If
   i = i + 1
   Input #Canal, Dato(i)
Loop
Close
Labelfinal = Labelfinal & Dato(i) & vbCrLf
Labelfinal = Labelfinal & vbCrLf &
"Finalizada la extracción el total de máquinas
es" & i
End Sub
```

Código (versiones más recientes de VB):

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1

Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

        Dim i As Integer
        Dim Dato(50) As Single
        Me.Text = "Maquinaria"

Labelfinal.TextAlign = ContentAlignment.MiddleCenter
Labelfinal.Font = New Font("Arial", 10,
FontStyle.Bold)
Labelfinal.Text = ""
i = 1
Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Toshiba\Desktop\maqui
naria.dat", False)

Dato(i) = myFileToRead.ReadLine()
Labelfinal.Text = "Números de serie de palas
cargadoras" & vbCrLf & vbCrLf

Do While Dato(i) <> 63287
Labelfinal.Text = Labelfinal.Text & Dato(i) & vbCrLf

If Dato(i) = 78345 Then
Labelfinal.Text = Labelfinal.Text & vbCrLf & "Números
de serie de retroexcavadoras" & vbCrLf & vbCrLf
End If

If Dato(i) = 33287 Then
Labelfinal.Text = Labelfinal.Text & vbCrLf & "Números
de serie de dumpers" & vbCrLf & vbCrLf
End If

i = i + 1
Dato(i) = myFileToRead.ReadLine()
Loop

Labelfinal.Text = Labelfinal.Text & Dato(i) & vbCrLf
Labelfinal.Text = Labelfinal.Text & vbCrLf &
"Finalizada la extracción el total de máquinas es" & i
myFileToRead.Close()
End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

El resultado gráfico con los datos de prueba indicados será este:



EJERCICIO RESUELTO Nº 2 EJEMPLO DE USO DE FOR … NEXT, CONSTANTES vbYes Y vbNo, MSGBOX…

Generar un programa que resuelva el siguiente problema: calcular el sumatorio de los 10 primeros números enteros positivos y según decida el usuario calcular o no la suma de los *20, 30, 40..., n – 10, n* enteros positivos.

SOLUCIÓN

Código (versiones menos recientes de VB):

```
'Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim i%, j As Integer
Dim Respuesta As Integer
Dim Suma As Integer
Private Sub Form_Load()
Show
j = 1
Respuesta = vbYes
Do While Respuesta = vbYes
   For i = j To j + 9
      Suma = Suma + i
   Next i
   j = j + 10
   Respuesta = MsgBox("La suma de los " & j - 1 & " primeros números da " & Suma & vbCrLf & _
   "¿Quiere continuar?", vbYesNo, "¿Continuar?")
Loop
End Sub
```

Código (versiones más recientes de VB):

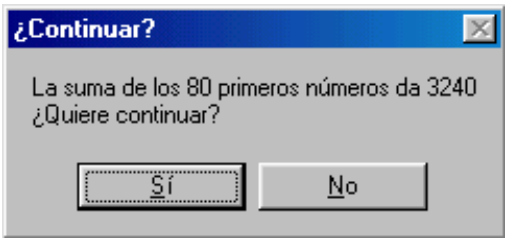
```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim i, j As Integer
        Dim Respuesta As Integer
        Dim Suma As Integer
        j = 1
        Respuesta = vbYes
Do While Respuesta = vbYes
For i = j To j + 9
         Suma = Suma + i
Next i
j = j + 10
Respuesta = MsgBox("La suma de los " & j - 1 & "
primeros números da " & Suma & vbCrLf & _
"¿Quiere continuar?", vbYesNo, "¿Continuar?")
        Loop
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

En vez de usar una variable string para almacenar la respuesta, usamos los botones predeterminados de Visual Basic y las constantes vbYes (sí) y vbNo (no).

Debemos conocer la ruta en que se ubica el archivo de datos y escribirla correctamente.

Gráficamente el resultado será similar a este:



Próxima entrega: CU00334A-2

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

El objetivo de estos ejercicios es aplicar conocimientos sobre "Herramientas básicas para la programación" a un lenguaje como es Visual Basic. Para su realización debemos tener conocimientos básicos sobre algoritmia, uso de bucles o estructuras iterativas y apertura y cierre de archivos.

EJERCICIO RESUELTO EJEMPLO DE USO DE ARRAYS ESTÁTICOS, DO WHILE … LOOP, IF … THEN, USO DE FICHEROS…

Desarrollar un programa que permita resolver el siguiente problema. Una librería dispone de un archivo donde se guarda información relativa a un lote de libros estructurada de la siguiente manera:

Dato(1) = [Número de ISBN, 10 dígitos]

Dato(2) = [Precio de venta al público, en euros]

Dato(3) = [Señalero, – 33 Libro de menos de un año de antigüedad
– 66 Libro de uno a tres años de antigüedad
– 99 Libro de más de tres años de antigüedad]

Dato(4) = [Número de ISBN]

.
.
.

Dato(n – 1) = [Señalero – 33, – 66, – 99]

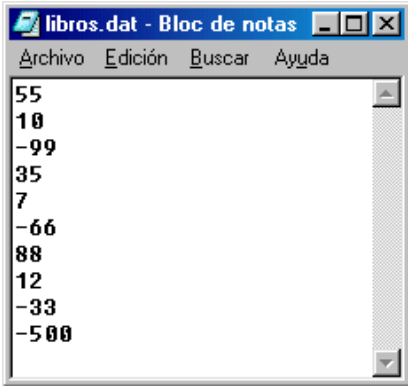
Dato(n) = [Señalero – 500, Fin del lote]

Se desea desarrollar un programa que permita:

a) Consultar el precio de un libro dado su ISBN.
b) Conocer el precio del lote de libros en conjunto.

Teniendo en cuenta lo siguiente: si el libro tiene menos de un año de antigüedad su precio coincide con el registrado. Si tiene entre uno y tres años deberá aplicarse un descuento del 15%. Si tiene más de tres años deberá aplicarse una rebaja del 25%.

Nota: considerar que el número total de datos es igual o inferior a 50, y que se encuentran en el fichero libros.dat. Crear un botón para poder introducir el isbn de un libro y otro botón para poder calcular el precio del lote. En un archivo al que llamaremos libros.dat pondremos una simulación de datos como la siguiente:



© 2006-2029 aprenderaprogramar.com

SOLUCIÓN

Código versiones menos recientes de VB:

```
Rem Curso VB aprenderaprogramar.com
Option Explicit
Dim i%, Canal As Integer
Dim Dato(50) As Single
Dim ISBN!, Precio!, Valorlote As Single
Dim Encontrado As Boolean

Private Sub Form_Load()
Show
Form1.Caption = "Libros"
CommandLibro.Caption = "Precio de un
libro"
CommandLote.Caption = "Precio del lote"
LabelResultado.Alignment = 2
LabelResultado.FontBold = True
End Sub

Private Sub CommandLibro_Click()
Encontrado = False
ISBN = InputBox("¿Cuál es el ISBN del
libro?", "ISBN")
Canal = FreeFile
i = 1
Open "C:\libros.dat" For Input As Canal
Input #Canal, Dato(i)
Do While Dato(i) <> -500
   If Dato(i) = ISBN Then
      Input #Canal, Dato(i + 1)
      Input #Canal, Dato(i + 2)
      Precio = Dato(i + 1)
      If Dato(i + 2) = -66 Then
         Precio = 0.85 * Precio
      End If

      If Dato(i + 2) = -99 Then
         Precio = 0.75 * Precio
      End If
      LabelResultado = "El precio del libro
con ISBN " & ISBN & " es de " & Precio
      Encontrado = True
   End If
   i = i + 3
   Input #Canal, Dato(i)
Loop
Close
If Encontrado = False Then
   LabelResultado = "No se ha
encontrado el ISBN deseado"
End If
End Sub
```

Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim Dato(50) As Single
    Dim ISBN, Precio, Valorlote As Single
    Dim Encontrado As Boolean

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.Text = "Libros"
        CommandLibro.Text = "Precio de un libro"
        CommandLote.Text = "Precio del lote"
        LabelResultado.TextAlign =
ContentAlignment.MiddleCenter
        LabelResultado.Font = New Font("Arial", 10,
FontStyle.Bold)
        LabelResultado.Text = ""
    End Sub

    Private Sub CommandLibro_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CommandLibro.Click
        Dim i As Integer
        Encontrado = False
        ISBN = InputBox("¿Cuál es el ISBN del libro?",
"ISBN")
        Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Toshiba\Desktop\libro
s.dat", False)
        i = 1
        Dato(i) = myFileToRead.ReadLine()
        Do While Dato(i) <> -500
            If Dato(i) = ISBN Then
                Dato(i + 1) = myFileToRead.ReadLine()
                Dato(i + 2) = myFileToRead.ReadLine()
                Precio = Dato(i + 1)
                If Dato(i + 2) = -66 Then
                    Precio = 0.85 * Precio
                End If
                If Dato(i + 2) = -99 Then
                    Precio = 0.75 * Precio
                End If
                LabelResultado.Text = "El precio del
libro con ISBN " & ISBN & " es de " & Precio
                Encontrado = True
            End If
            i = i + 3
            Dato(i) = myFileToRead.ReadLine()
        Loop
        myFileToRead.Close()
        If Encontrado = False Then
            LabelResultado.Text = "No se ha encontrado
el ISBN deseado"
         End If
     End Sub
```

© 2006-2029 aprenderaprogramar.com

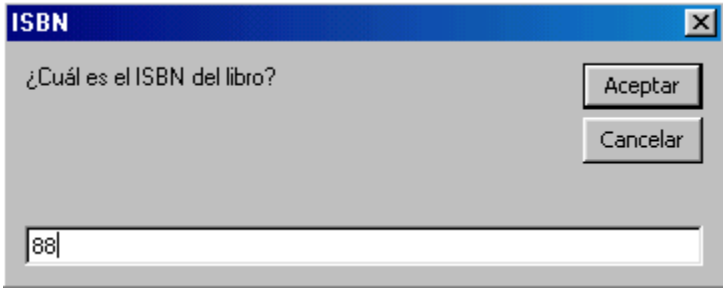
Código versiones menos recientes de VB:

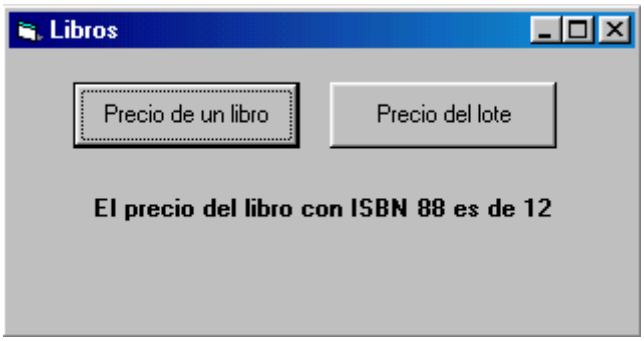
```
Private Sub CommandLote_Click()
Valorlote = 0
Canal = FreeFile
i = 1
Open "C:\libros.dat" For Input As Canal
Input #Canal, Dato(i)
Do While Dato(i) <> -500
   Input #Canal, Dato(i + 1)
   Input #Canal, Dato(i + 2)
   Precio = Dato(i + 1)
   If Dato(i + 2) = -66 Then
      Precio = 0.85 * Precio
   End If
   If Dato(i + 2) = -99 Then
      Precio = 0.75 * Precio
   End If
   Valorlote = Valorlote + Precio
   i = i + 3
   Input #Canal, Dato(i)
Loop
Close
LabelResultado = "El precio del lote es "
& Valorlote
End Sub
```

Código versiones más recientes de VB:

```
Private Sub CommandLote_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CommandLote.Click
   Dim i As Integer
   Valorlote = 0
Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Toshiba\Desktop\libros.dat
", False)
        i = 1
        Dato(i) = myFileToRead.ReadLine()
        Do While Dato(i) <> -500
            Dato(i + 1) = myFileToRead.ReadLine()
            Dato(i + 2) = myFileToRead.ReadLine()
            Precio = Dato(i + 1)
            If Dato(i + 2) = -66 Then
                Precio = 0.85 * Precio
            End If
            If Dato(i + 2) = -99 Then
                Precio = 0.75 * Precio
            End If
            Valorlote = Valorlote + Precio
            i = i + 3
            Dato(i) = myFileToRead.ReadLine()
        Loop
        myFileToRead.Close()
        LabelResultado.Text = "El precio del lote es "
& Valorlote
    End Sub
End Class
```

Gráficamente:





Próxima entrega: CU00335A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

FINAL DE ARCHIVO CON VISUAL BASIC. EJEMPLO.

En muchos casos no sabemos el número de datos que existen en un archivo y necesitaremos ir extrayendo datos hasta llegar al final del archivo. Para ello vamos a proponer dos opciones:

a) Para versiones menos recientes de Visual Basic usaremos EOF (End Of File). EOF es un parámetro booleano útil para facilitar el cierre de bucles de extracción de datos desde archivo. En Visual Basic EOF es una función que al ser invocada devuelve un valor tipo Integer/Booleano que es 0 (False) mientras extraemos datos que existen desde un archivo y -1 (True) al llegar al final del archivo.

Si se trata de extraer mayor número de datos del que existen en un archivo aparece el error "La entrada de datos ha sobrepasado el final del archivo". Para evitarlo utilizaremos la función *EOF* con cualquiera de estas sintaxis:

```
Do While Not EOF(Canal)
```

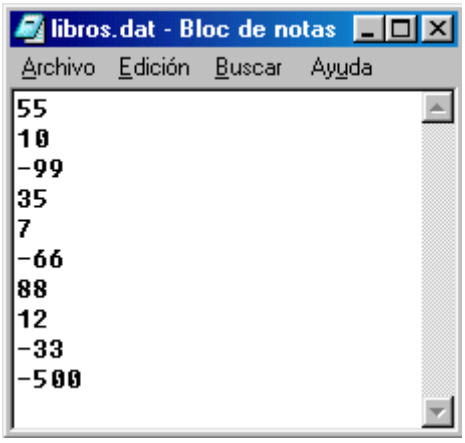
```
Do While EOF(Canal) = False
```

b) Para versiones más recientes de Visual Basic iremos extrayendo datos hasta que al llegar al final de archivo se produzca que la extracción toma el valor predefinido de Visual Basic “Is Nothing” = true. “Is Nothing” es un indicador booleano de que un valor extraído es vacío o sin contenido. Mientras existen datos en el archivo y se van extrayendo Is Nothing valdrá false. Cuando se llegue al final de archivo Is Nothing tomará valor true. La sintaxis será:

```
Do While extraccion Is Nothing = False
```

Donde extraccion simboliza un nombre de variable donde estamos extrayendo los datos leídos desde el archivo.

Consideremos un archivo libros.dat creado previamente donde tenemos 10 datos numéricos.



© 2006-2029 aprenderaprogramar.com

La extracción de datos hasta llegar a final de archivo se haría de la siguiente manera:

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit

Dim i%, Canal As Integer
Dim Dato(50) as Single

Private Sub Form_Load()
Show
LabelResultado = "Datos extraidos de
archivo: "
Canal = FreeFile
Open "C:\libros.dat" For Input As Canal

Do While EOF(Canal) = False
   i = i + 1
   Input #Canal, Dato(i)
   LabelResultado = LabelResultado &
Dato(i) & " , "
Loop
Close
End Sub
```

Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
Dim i As Integer
Dim Dato(50) As Single
Dim lineaExtraida As String
LabelResultado.Text = "Datos extraidos de archivo: "
Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Asus\Desktop\libros.dat"
, False)
 lineaExtraida = myFileToRead.ReadLine()
 Do While lineaExtraida Is Nothing = False
     Dato(i) = lineaExtraida
     LabelResultado.Text = LabelResultado.Text & Dato(i)
& " , "
     i = i + 1
     lineaExtraida = myFileToRead.ReadLine()
 Loop
 myFileToRead.Close()
End Sub
End Class
```

EJERCICIO USO DE FINAL DE ARCHIVO

Resolver este problema:

En un fichero tenemos las notas de un grupo de alumnos de una clase en un array Nota(1), Nota(2), …, Nota(n), establecidas entre 0 y 10. Se desea desarrollar un programa que determine la nota media extrayendo todos los datos existentes hasta llegar al final de archivo.

Nota: Considerar que el número de datos es igual o inferior a 50.

Vamos a considerar que el archivo con los datos es notas.dat y que contiene los 35 valores siguientes cada uno en una línea (aquí los escribimos separados por guiones para no ocupar 35 líneas):

6 – 7 – 8 – 5 – 6 – 4 – 7 – 8 – 7 – 10 – 0 – 5 – 6 – 7 – 6 – 5 – 0 – 7 – 8 – 10 – 2 – 3 – 4 – 5 – 6 – – 7 – 6 – 5 – 4 – 10 – 4 – 3 – 2 – 3 – 4

Necesitamos conocer la ruta donde está ubicado el archivo e indicarla correctamente.

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN

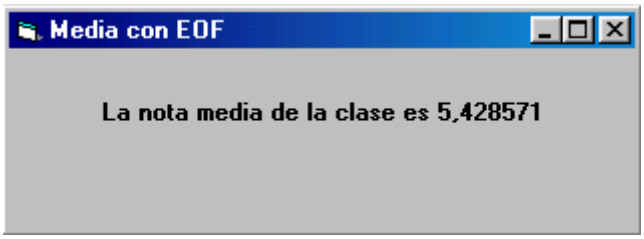
Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com

'[Media con Fin de Archivo]

Option Explicit
Dim i%, Canal As Integer
Dim Nota(50) As Single
Dim Media!, Suma As Single

Private Sub Form_Load()
Show
Form1.Caption = "Media con lectura
hasta fin de archivo"
Canal = FreeFile
Open "C:\notas.dat" For Input As Canal
Do While EOF(Canal) = False
   i = i + 1
   Input #Canal, Nota(i)
   Suma = Suma + Nota(i)
Loop
Close
Media = Suma / i
LabelResultado.Alignment = 2
LabelResultado.FontBold = True
LabelResultado = "La nota media de la
clase es " & Media
End Sub
```

Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim i As Integer
        Dim Nota(50) As Single
        Dim lineaExtraida As String
        Dim Media, Suma As Single

Me.Text = "Media con lectura datos hasta fin de archivo"
Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Asus\Desktop\notas.dat"
, False)

        lineaExtraida = myFileToRead.ReadLine()
        Do While lineaExtraida Is Nothing = False
            i = i + 1
            Nota(i) = lineaExtraida
            Suma = Suma + Nota(i)
            lineaExtraida = myFileToRead.ReadLine()
        Loop
        myFileToRead.Close()
        Media = Suma / i
LabelResultado.TextAlign = ContentAlignment.MiddleCenter
LabelResultado.Font = New Font("Arial", 10,
FontStyle.Bold)
LabelResultado.Text = "La nota media de la clase es " &
Media
    End Sub
End Class
```

El resultado del programa:



Próxima entrega: CU00336A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

MODIFICACIÓN DIRECTA DEL FLUJO DE PROGRAMAS CON VISUAL BASIC.

Las recomendaciones para el uso de instrucciones de control directo del flujo de programas se resumen en una palabra: prudencia. Si no tienes cuidado con el uso de instrucciones que modifican el curso natural de los programas, puedes ocasionar el mal funcionamiento de los mismos o comprobar que has generado un código de poca calidad y difícil seguimiento.

INSTRUCCIÓN FINALIZAR. END, UNLOAD, EXIT SUB.

Con Visual Basic disponemos de distintas variantes relacionadas con Finalizar.

a) End: instrucción que da lugar a la detención instantánea de un programa. No se recomienda cerrar un programa directamente a través de End, ya que hay instrucciones previstas para un cierre normal. Podríamos considerar que End supone "apagar" el programa como si hubiera un corte de energía eléctrica, mientras que la descarga del programa equivale al cierre de un sistema operativo a través de la opción "Cerrar". Para la finalización normal de un programa de Visual Basic debemos descargar todos los formularios con que estemos trabajando (lo cual supone que se produzcan los eventos QueryUnload y Unload) antes de disponer la instrucción End.

b) Unload ó Close (según versiones): instrucción que da lugar a la descarga de la memoria de formularios o controles, lo cual no necesariamente significa que el programa acabe, ya que el código asociado al módulo del formulario permanece en memoria y pueden mostrarse otro tipo de ventanas o incluso volver a cargarse el formulario si así está establecido en el código.

c) Exit Sub: da lugar a que se salga del procedimiento que se está ejecutando, continuando el flujo del programa como si se hubiera llegado al final del procedimiento.

Veamos ejemplos de código:

a) Código asociado a un botón que da lugar al cierre del programa.

Versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Private Sub ButtonCerrar_Click()
MsgBox("Se va a cerrar el
formulario y el programa")
Unload Form1
End
End Sub
```

Versiones más recientes de VB:

```
REM Curso VB aprenderaprogramar.com
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
     MsgBox("Se va a cerrar el formulario y el programa")
     Me.Close()
     End
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

b) Código que da lugar a la salida de un procedimiento

Versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Private Sub ButtonCerrar_Click()
Dim condición As Integer
condición = MsgBox("¿Desea
continuar?", vbYesNo)
If condición = vbNo Then Exit Sub
MsgBox("Ha elegido continuar")
End Sub
```

Versiones más recientes de VB:

```
REM Curso VB aprenderaprogramar.com
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim condición As Integer
        condición = MsgBox("¿Desea continuar?", vbYesNo)
        If condición = vbNo Then Exit Sub
        MsgBox("Ha elegido continuar")
    End Sub
End Class
```

En el primer ejemplo, se muestra un mensaje y se cierra el programa. En el segundo código, si el usuario elige que desea continuar se muestra el mensaje "Ha elegido continuar" y prosigue la ejecución del módulo. Si elige que no desea continuar se ejecuta Exit Sub. Se produce el salto a la última línea del procedimiento (End Sub) y el flujo del programa continua por donde proceda.

EJERCICIO

Crear un programa en Visual Basic a partir del siguiente pseudocódigo, que constituye un ejemplo de aplicación de la Instrucción Finalizar.

```
1. Inicio [Finaliza si hay problemas. Curso Visual Basic aprenderaprogramar.com]
        2. Mientras Numero < 100 ó Numero > 200 Hacer
                Mostrar "Por favor, introduzca un número comprendido entre 100 y 200"
                Pedir Numero
                i = i + 1
                Si i > 19 Entonces
                        Mostrar "Hay un problema para continuar y el programa se va a cerrar. Motivo:
                                20 procesos nulos. Consulte el manual del programa"
                        Finalizar
                FinSi
            Repetir
        3. Mostrar "Gracias por introducir una entrada válida"
        4. [Aquí continuaría el programa con diferentes procesos]
5. Fin
```

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN

Código para versiones menos recientes de Visual Basic:

```
'[Curso VB aprenderaprogramar.com]
Option Explicit
Dim i As Integer
Dim Número As Single
Private Sub Form_Load()
Show
Form1.Caption = "Ejemplo Finalizar"
Do While Número < 100 Or Número > 200
   Número = InputBox("Introduzca un número comprendido entre 100 y 200")
   i = i + 1
   If i > 19 Then
      MsgBox("Hay un problema para continuar y el programa se va a" & _
      "cerrar. Motivo: 20 procesos nulos. Consulte el manual del programa")
      Unload Form1
      End
   End If
Loop
Label1 = "Gracias por introducir una entrada válida"
'[Aquí continuaría el programa con diferentes procesos]
End Sub
```

Código para versiones más recientes de Visual Basic:

```
REM Curso VB aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        Dim i As Integer
        Dim Número As Single
        Me.Text = "Ejemplo Finalizar"
        Do While Número < 100 Or Número > 200
            Número = InputBox("Introduzca un número comprendido
entre 100 y 200")
            i = i + 1
            If i > 19 Then
                MsgBox("Hay un problema para continuar y el
programa se va a" & _
                "cerrar. Motivo: 20 procesos nulos. Consulte el
manual del programa")
                Me.Close()
                End
            End If
        Loop
        Label1.Text = "Gracias por introducir una entrada válida"
        '[Aquí continuaría el programa con diferentes procesos]
    End Sub
End Class
```

Próxima entrega: CU00337A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

SALIR DE UN BUCLE FOR CON EXIT FOR EN VISUAL BASIC

Cuando realizamos un esquema de programa usando pseudocódigo podemos reflejar la salida de un bucle For (Desde … Siguiente) con la instrucción SalirDesde. La sintaxis a emplear en Visual Basic es Exit For dentro de cualquier bucle For ... Next. Cuando existe anidamiento de bucles, se produce la salida del bucle más interno al inmediato superior. Si se escribe un Exit For fuera de un bucle aparece el mensaje: "Error de compilación. Exit For no está dentro de For ... Next".

EJERCICIO RESUELTO CON EXIT FOR

Transformar en código el ejemplo de aplicación de la Instrucción SalirDesde que se muestra a continuación en forma de pseudocódigo, considerando que Numerodatos tiene un valor fijo de 35, y que los datos están el archivo notas.dat, que habrá sido creado previamente.

Pseudocódigo ejemplo de uso de SalirDesde (Exit For):

```
1. Inicio [Algoritmo Media curso VB aprenderaprogramar.com]
        2. Leer Numerodatos [Establece el nº de datos a extraer]
        3. [El dato esperado en fichero debe valer entre 0 y10]
        4. Desde i = 1 hasta Numerodatos Hacer
                Leer Dato(i)
                Si Dato(i) < 0 ó Dato(i) >10 Entonces [Dato no válido]
                        Mostrar “No se pudo completar la extracción de datos.
                                Motivo: dato no válido. Revise archivo y vuelva a intentarlo”
                        Fallolectura = Verdadero [Interruptor]
                        SalirDesde
                FinSi
                Suma = Suma + Dato(i)
           Siguiente
        5. Si Fallolectura = Falso Entonces
                Media = Suma / Numerodatos
                Mostrar “La media es”, Media
           Sino
                Mostrar “Gracias por utilizar este programa”
           FinSi
6. Fin
```

SOLUCIÓN AL EJERCICIO CON EXIT FOR

Como paso previo en tiempo de diseño crearemos un label para mostrar los resultados al que denominaremos LabelResultado así como un botón que el usuario debe pulsar para leer los datos desde el archivo.

© 2006-2029 aprenderaprogramar.com

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com

Option Explicit
Dim Canal%, i As Integer
Dim Dato(35) As Single
Dim Fallolectura As Boolean
Dim Suma!, Media As Single

Private Sub Form_Load()
Show
Form1.Caption = "Notas"
CommandCalcular.Caption = "Leer
archivo"
End Sub

Private Sub CommandCalcular_Click()
LabelResultado.Alignment = 2
LabelResultado.FontBold = True
'[Apertura del archivo]
Canal = FreeFile
Open "C:\notas.dat" For Input As Canal
For i = 1 To 35
   '[Lectura de datos desde archivo]
   Input #Canal, Dato(i)
   If Dato(i) < 0 Or Dato(i) > 10 Then
'[Dato no válido]
      LabelResultado = "No se pudo
completar la extracción de datos. Motivo:
dato no válido. Revise archivo y vuelva a
intentarlo"
      Fallolectura = True  '[Interruptor]
      Exit For
   End If
   Suma = Suma + Dato(i)
Next i
Close Canal '[Cerramos el archivo]
If Fallolectura = False Then
   Media = Suma / 35
   LabelResultado = "La media es " &
Media
Else
   LabelResultado = LabelResultado &
vbCrLf & "Gracias por utilizar este
programa"
End If
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com

Option Explicit On

Public Class Form1
    Dim Dato(35) As Single
    Dim Fallolectura As Boolean
    Dim Suma, Media As Single

Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Me.Text = "Notas"
        ButtonCalcular.Text = "Leer archivo"
    End Sub


    Private Sub ButtonCalcular_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ButtonCalcular.Click

        Dim i As Integer

LabelResultado.TextAlign = ContentAlignment.MiddleCenter
LabelResultado.Font = New Font("Arial", 10,
FontStyle.Bold)
LabelResultado.Text = ""
Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Asus\Desktop\notas.da
t", False)

        For i = 1 To 35
            '[Lectura de datos desde archivo]
            Dato(i) = myFileToRead.ReadLine()
            If Dato(i) < 0 Or Dato(i) > 10 Then '[Dato
no válido]
                LabelResultado.Text = "No se pudo
completar la extracción de datos. Motivo: " & _
"dato no válido. Revise archivo y vuelva a intentarlo"
                Fallolectura = True  '[Interruptor]
                Exit For
            End If
            Suma = Suma + Dato(i)
        Next i
        myFileToRead.Close()

        If Fallolectura = False Then
        Media = Suma / 35
        LabelResultado.Text = "La media es " & Media
        Else
        LabelResultado.Text = LabelResultado.Text &
vbCrLf & "Gracias por utilizar este programa"
        End If
    End Sub
End Class
```

Para comprobar el funcionamiento del programa, lo ejecutaremos con un archivo de datos válidos y después modificaremos un dato para hacerlo no válido, con lo cual deberá mostrarse el mensaje "No se pudo completar la extracción de datos. Motivo: dato no válido. Revise archivo y vuelva a intentarlo".

© 2006-2029 aprenderaprogramar.com

SALIR DE UN BUCLE DO…LOOP WHILE CON EXIT DO EN VISUAL BASIC

La sintaxis a emplear es Exit Do dentro de cualquier bucle Do ... Loop While. Cuando existe anidamiento de bucles, se produce la salida del bucle más interno al inmediato superior. Si se escribe un Exit Do fuera de un bucle aparece el mensaje: "Error de compilación. Exit Do no está dentro de Do ... Loop".

INSTRUCCIÓN HACER … REPETIR CON DO … LOOP Y SALIRHACER CON EXIT DO

Dado que con Visual Basic Mientras ... Hacer se construye con el mismo comando (Do) que Hacer ... Repetir Mientras, la instrucción de salida de este tipo de bucles es siempre Exit Do, con las consideraciones que ya hemos hecho. La construcción Hacer...Repetir se consigue usando Do...Loop sin condicionantes a la entrada ni a la salida. Esto nos obliga, necesariamente, a que exista una condición de salida interna y a que exista un Exit Do dentro del bucle. Consideremos el siguiente ejemplo:

```
Do
   i = i + 1
   Label1 = Label1 & i
   If i > 5 Then Exit Do
Loop
```

En caso de un fallo del contador o de la condición de salida se producirá un bucle infinito. Es poco recomendable usar construcciones tipo Do ... Loop como elemento habitual de programación. Lo más razonable es usarla únicamente en casos excepcionales y en líneas generales prescindir de ella.

EJERCICIO RESUELTO CON USO DE DO … LOOP Y EXIT DO

Transformar en código el pseudocódigo que se muestra a continuación como ejemplo de aplicación de SalirHacer y Hacer ... Repetir.

Nota: considerar que el valor de SecretCode se encuentra en el archivo code.dat creado previamente.

Pseudocódigo ejemplo de uso de SalirHacer (Exit Do) y de Hacer ... Repetir (Do…Loop).

```
1. Inicio [Mensaje si se acierta número secreto curso VB aprenderaprogramar.com]
      2. Leer Secretcode [Valor del número secreto]
      3. Hacer
            Mostrar "Introduzca número secreto"
            Pedir Numero
            Si Numero = Secretcode Entonces
                  Mostrar "Bienvenido. Su clave para operar será 685297"
                  SalirHacer
            SiNo
                  Mostrar "Número erróneo"
                  i = i + 1
                  Si i > 19 Entonces
                        Mostrar "Excedido número de pruebas disponibles. Acceso denegado"
                        SalirHacer
                  FinSi
            FinSi
         Repetir
4. Fin
```

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN AL EJERCICIO CON USO DE DO … LOOP Y EXIT DO

Código versiones menos recientes VB:

```
'[Curso VB aprenderaprogramar.com]
'[Mensaje si se acierta número secreto]

Option Explicit
Dim Canal%, i As Integer
Dim SecretCode!, Número As Single

Private Sub Form_Load()
Show
LabelResultado.Alignment = 2
LabelResultado.FontBold = True
'[Apertura del archivo]
Canal = FreeFile
Open "C:\code.dat" For Input As Canal
'[Lectura de datos desde archivo]
Input #Canal, SecretCode
Close Canal '[Cerramos el archivo]
Do
Número = InputBox("Introduzca número secreto")
If Número = SecretCode Then
    LabelResultado = "Bienvenido. Su clave para operar será 685297"
    Exit Do
Else
    MsgBox "Número erróneo"
    i = i + 1
If i > 19 Then
LabelResultado = "Excedido número de pruebas disponibles. Acceso denegado"
Exit Do
End If
End If
Loop
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

        Dim SecretCode, Número As Single
        Dim i As Integer
        Me.Text = "Secret Code"

LabelResultado.TextAlign = ContentAlignment.MiddleCenter
LabelResultado.Font = New Font("Arial", 10, FontStyle.Bold)
LabelResultado.Text = ""
'[Apertura del archivo]

Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Asus\Desktop\code.dat"
, False)
        '[Lectura de datos desde archivo]
        SecretCode = myFileToRead.ReadLine()
        myFileToRead.Close()

        Do
        Número = InputBox("Introduzca número secreto")
        If Número = SecretCode Then
        LabelResultado.Text = "Bienvenido. Su clave
para operar será 685297"
        Exit Do
        Else
        MsgBox("Número erróneo")
        i = i + 1
            If i > 19 Then
            LabelResultado.Text = "Excedido número de
pruebas disponibles. Acceso denegado"
            Exit Do
            End If
        End If

        Loop
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

INSTRUCCIÓN GOTO CON VISUAL BASIC

El uso de la instrucción IrA (GoTo) no debe contemplarse en la práctica habitual de la programación. La sintaxis a emplear es:

```
GoTo [número de línea]
```

```
GoTo [etiqueta]
```

Para ejecutar un GoTo no es necesario que todas las líneas estén identificadas, pero sí es necesario que la línea de destino esté numerada o etiquetada. No se puede hacer un GoTo a un procedimiento distinto del que se esté ejecutando en ese momento.

EJERCICIO RESUELTO CON USO DE GOTO

Generar el código correspondiente al pseudocódigo que se muestra a continuación, considerando que Numerodatos es un valor fijo de 35, y que los datos se encuentran en el fichero notas.dat, que se habrá creado previamente.

Pseudocódigo ejemplo de uso de IrA (GoTo).

```
1. Inicio [Ejemplo de IrA curso VB aprenderaprogramar.com]
        2. Leer Numerodatos [Establece el nº de datos a extraer]
        3. [El dato esperado en fichero debe valer entre cero y diez]
        4. Desde i = 1 hasta Numerodatos Hacer
                Leer Dato(i)

                Si Dato(i) < 0 ó Dato(i) > 10 Entonces [Dato no válido]
                        IrA 99 [Envío a Gestión de Errores]
                FinSi
                Suma = Suma + Dato(i)
           Siguiente
        5. Media = Suma / Numerodatos
        6. Mostrar "La media es", Media
        7. Finalizar
        99. [Gestión de Errores]
                99.1 Mostrar "No se pudo completar la extracción de datos. Motivo:
                                dato no válido. Revise archivo y vuelva a intentarlo"
                99.2 Mostrar "Gracias por utilizar este programa"
100. Fin
```

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN AL EJERCICIO CON USO DE GOTO

Código versiones menos recientes VB:

```
'[Curso VB aprenderaprogramar.com]
'[Ejemplo de IrA]

Option Explicit
Dim Canal As Integer
Dim i As Integer
Dim Dato(35) As Single
Dim Suma!, Media As Single

Private Sub Form_Load()
Show
LabelResultado.Alignment = 2
LabelResultado.FontBold = True
'[Apertura del archivo]
Canal = FreeFile
Open "C:\notas.dat" For Input As Canal
For i = 1 To 35
   '[Lectura de datos desde archivo]
   Input #Canal, Dato(i)
   If Dato(i) < 0 Or Dato(i) > 10 Then
'[Dato no válido]
      GoTo 99 '[Envío a gestión de errores]
   End If
   Suma = Suma + Dato(i)
Next i
Media = Suma / 35
MsgBox("La media es " & Media)
Close Canal '[Cerramos el archivo]
Unload Form1
End
99 '[Gestión de errores]
LabelResultado = "No se pudo completar la
extracción de datos. Motivo: dato no válido.
Revise archivo y vuelva a intentarlo" &
vbCrLf & "Gracias por utilizar este
programa"
Close Canal  '[Cerramos el archivo]
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

        Dim i As Integer
        Dim Dato(35) As Single
        Dim Suma, Media As Single

LabelResultado.TextAlign = ContentAlignment.MiddleCenter
LabelResultado.Font = New Font("Arial", 10,
FontStyle.Bold)
        LabelResultado.Text = ""
        '[Apertura del archivo]

Dim myFileToRead As New System.IO.StreamReader
("C:\Users\Asus\Desktop\notas.dat", False)

        For i = 1 To 35
            '[Lectura de datos desde archivo]
            Dato(i) = myFileToRead.ReadLine()
            If Dato(i) < 0 Or Dato(i) > 10 Then
'[Dato no válido]
                GoTo 99 '[Envío a gestión de errores]
            End If
            Suma = Suma + Dato(i)
        Next i
        Media = Suma / 35
        MsgBox("La media es " & Media)
        myFileToRead.Close()
        Me.Close()
        End
99:     '[Gestión de errores]
        LabelResultado.Text = "No se pudo completar
la extracción de datos. Motivo: dato no válido.
Revise archivo y vuelva a intentarlo" & vbCrLf &
"Gracias por utilizar este programa"
        myFileToRead.Close()
    End Sub
End Class
```

Próxima entrega: CU00338A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

PROGRAMACIÓN DE MÓDULOS. TIPOS. PROCEDIMIENTOS SUB, FUNCIONES FUNCTION.

Visual Basic, como no podía ser de otra manera, está orientado a la programación modular. Ya hemos comentado que puede haber cierta confusión terminológica: el concepto de módulo que hemos usado en cursos de bases de programación de aprenderaprogramar.com no coincide con el concepto de módulo en la terminología de Visual Basic. Vamos a hacer una clasificación libre tratando de conectar el Visual Basic con el desarrollo que hemos realizado en otros cursos.

DESCRIPCIÓN	TERMINOLOGÍA DE VISUAL BASIC
Formulario: ventana de formulario, objetos sobre el formulario, código	Módulo de formulario (fichero *.frm ó *.vb, según versiones)
Código de declaraciones y definiciones de procedimientos, tipos y datos con carácter público para otros módulos	Módulo estándar (fichero *.bas ó *.vb, según versiones)
Código de definición de una clase de objeto	Módulo de clase (Fichero *.cls ó *.vb, según versiones)
Módulo de código que se ejecuta cuando tiene lugar un evento determinado	Procedimiento conducido por eventos (Sub)
Módulo de código que se ejecuta cuando es llamado desde algún punto del programa	Procedimientos generales (Sub)
Módulo de código que se ejecuta cuando es llamado desde algún punto del programa y devuelve un valor	Procedimiento función (Function)

Dado que el código contenido en un módulo estándar de Visual Basic es accesible desde distintos formularios del programa, será ventajoso colocar en este módulo todo lo que queramos disponer como "código compartido".

Buscando analogías con el desarrollo que hacemos en los cursos de pseudocódigo de aprenderaprogramar.com, usaremos el evento que se produce cuando ordenamos la ejecución de nuestros programas (carga del formulario o Form_Load) para disponer en él el código del "algoritmo principal" o guía del programa y el resto del código irá ordenado en procedimientos conducidos por eventos, procedimientos generales ó procedimientos función.

Habíamos dicho que un módulo no se ejecuta hasta que es llamado a ejecutarse desde el algoritmo principal de acuerdo con la sintaxis de pseudocódigo:

```
Llamar [Nombre del Módulo]
```

© 2006-2029 aprenderaprogramar.com

Sin embargo con programación guiada por eventos esto es sólo parcialmente cierto: un módulo puede ejecutarse por ser llamado desde algún punto del código pero también sin ser llamado desde el código, cuando tiene lugar un determinado evento.

La declaración de un **procedimiento general** se realizará mediante la sintaxis:

```
[Carácter Público o Privado] Sub [Nombre del procedimiento]([Parámetros])
```

EJEMPLO

```
Private Sub Calcular()
```

Todo procedimiento tiene un final indicado mediante End Sub, aunque se puede provocar una salida forzada usando la expresión Exit Sub.

El carácter público o privado se establece mediante las palabras clave Public o Private, que dan lugar a que dicho procedimiento pueda invocarse o no desde otros módulos. Sub indica que se está declarando un procedimiento Sub con un nombre determinado, y los paréntesis están destinados a contener parámetros requeridos por el procedimiento para ser invocado. El paso de parámetros lo veremos más adelante, por lo que consideraremos de momento procedimientos sin parámetro (paréntesis vacíos).

La declaración de un **procedimiento conducido por eventos** la realizaremos usando las listas desplegables de objetos y eventos que vimos cuando hablamos de botones (Buttons o Command Buttons). De esta manera, al seleccionar un objeto y un evento automáticamente nos aparecerá un código del tipo (según versiones habrá diferencias):

```
Private Sub Text1_Change()


End Sub
```

```
Private Sub TextBox1_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TextBox1.TextChanged


End Sub
```

Este es un procedimiento conducido por eventos que se invoca cuando el texto contenido en el TextBox cambia. No prestaremos atención de momento a los parámetros que puedan aparecer automáticamente.

La declaración de una **función** se realizará mediante la sintaxis:

```
[Carácter Público o Privado] Function [Nombre de la función]([Parámetros]) As [Tipo de dato]
```

© 2006-2029 aprenderaprogramar.com

EJEMPLO

```
Private Function Hipotenusa() As Single
.
.
.
End Function
```

Como hemos dicho, una función devuelve un valor, de ahí que especifiquemos un tipo de dato para ella. En caso de no especificarse, el tipo será Variant u Object (dependiendo de la versión) por defecto.

El flujo para una función sigue las mismas reglas que para un procedimiento general o conducido por eventos: al llegar a End Function el control vuelve a la sentencia inmediatamente posterior a la llamada efectuada. Se puede provocar la salida forzada de una función utilizando la expresión Exit Function.

Los procedimientos pueden insertarse en el programa en cualquier orden, aunque siempre será recomendable tratar de disponerlos en el mismo orden que está previsto que se ejecuten.

La llamada de un procedimiento general o conducido por eventos se realiza, cuando no hay parámetros que pasar, simplemente escribiendo su nombre, o bien usando Call [Nombre]. La llamada a una función se hará normalmente para obtener un valor o asignar un valor a una variable, en expresiones del tipo:

```
Label1.Text = Label1.Text & [Nombre de la función]()
Variable = [Nombre de la función]()
If [Nombre de la función]() > [Variable]
```

Hay que recordar siempre que una "función" ejecuta un código y devuelve un valor: podríamos decir que es un híbrido entre una variable y un procedimiento.

La llamada a un procedimiento desde sí mismo es posible, dando lugar a un anidamiento o recursión. Habrá de existir una condición que evolucione para dar lugar a la salida de la recursión, regresando el control del flujo a la instrucción posterior desde la que se autollamó el módulo. No vamos a desarrollar contenidos relativos a la recursión, pero si tienes interés en profundizar en esta técnica de programación te remitimos a profundizar en esta materia mediante otros cursos que se ofrecen en aprenderaprogramar.com.

Una llamada del algoritmo principal a sí mismo, que en su momento escribimos como Llamar Inicio, sería posible con Visual Basic, pero no vamos a entrar a detallar este tipo de cuestiones que raramente se usarán.

© 2006-2029 aprenderaprogramar.com

EJERCICIO

Transformar el siguiente programa "Comunicado" en código de Visual Basic.

1. **Inicio [PROGRAMA Comunicado curso Visual Basic aprenderaprogramar.com]**
   2. Mostrar “Comunicado de la empresa”
   3. Llamar Saludo
   4. Llamar Comunicado
   5. Llamar Despedida
6. **Fin**

Módulo Saludo

1. Mostrar “Con motivo de la celebración el próximo día 5 del Día Mundial del Medioambiente la empresa saluda a todos los empleados y les agradece el compromiso con el cuidado de la naturaleza”

FinMódulo

Módulo Comunicado

1. Mostrar “Con motivo de dicha conmemoración está previsto realizar un acto de plantación de árboles en los jardines del edificio central el próximo día 5 a las 12 del mediodía al que están todos invitados”

FinMódulo

Módulo Despedida

1. Mostrar “La empresa agradece su participación y les invita a sumarse al programa <<Empleados por una ciudad sostenible>>. Atentamente, El Director General”

FinMódulo

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN

Código versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
'Algoritmo principal

Private Sub Form_Load()
MsgBox("Comunicado de la empresa")
Saludo  'Invocación del procedimiento Saludo
Comunicado  'Invocación del procedimiento Comunicado
Despedida  'Invocación del procedimiento Despedida
End Sub

Private Sub Saludo()
MsgBox("Con motivo de la celebración el próximo día 5 del Día Mundial del Medioambiente")
End Sub

Private Sub Comunicado()
MsgBox("Se invita a todos los empleados a un acto de plantación de árboles a las 5 de la tarde en los jardines del edificio central")
End Sub

Private Sub Despedida()
MsgBox("La empresa agradece su participación")
End Sub
```

Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
MsgBox("Comunicado de la empresa")
Saludo()  'Invocación del procedimiento Saludo
Comunicado()  'Invocación del procedimiento
Comunicado
Despedida()  'Invocación del procedimiento
Despedida
    End Sub

    Private Sub Saludo()
        MsgBox("Con motivo de la celebración el
próximo día 5 del Día Mundial del
Medioambiente")
    End Sub

    Private Sub Comunicado()
        MsgBox("Se invita a todos los empleados
a un acto de plantación de árboles a las 5 de la
tarde en los jardines del edificio central")
    End Sub

    Private Sub Despedida()
        MsgBox("La empresa agradece su
participación")
    End Sub
End Class
```

El único interés de este ejercicio es reforzar la idea de estructura basada en algoritmo principal e invocación de módulos a la hora de construir nuestros programas.

Próxima entrega: CU00339A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

VARIABLES LOCALES, DE CLASE Y GLOBALES EN VISUAL BASIC.

Ya hemos dicho que existen distintas formas de declarar variables con Visual Basic, aunque hayamos venido utilizando principalmente una: la instrucción Dim. Vamos a ver ahora el concepto de variable local, de clase y global trasladado a este lenguaje.

En primer lugar, usaremos el término ámbito para referirnos a aquel conjunto de partes del programa en el que una variable es conocida. Así distinguiremos:

- **Variables con ámbito un procedimiento (Sub).**

Son declaradas dentro del procedimiento y sólo son conocidas dentro de él. Son declaradas al principio del procedimiento (cabecera del procedimiento) o en un punto intermedio del mismo. No puede invocarse la variable sin que antes se haya declarado. En general, para una mejor ordenación del programa siempre será preferible que las declaraciones sean en cabecera. Podemos llamarlas variables locales al procedimiento.

- **Variables con ámbito el módulo o la clase de formulario.**

Son declaradas en la cabecera del código asociado al formulario y no están dentro de un procedimiento o función (Sub) concreto. Podemos llamarlas variables locales al módulo de formulario ó en las versiones más recientes de Visual Basic la denominación es “variables de clase” o “atributos de la clase”, ya que en las versiones más recientes de Visual Basic decimos que el código se organiza en “clases”. Estas variables son conocidas por todos los procedimientos existentes en el código asociado al formulario.

- **Variables con ámbito todos los módulos o todas las clases del programa.**

Son declaradas dentro de un módulo estándar usando la declaración Public.

Prueba el siguiente código (izquierda, versiones menos recientes VB. Derecha: versiones más recientes):

```
'Curso VB aprenderaprogramar.com
Option Explicit

Private Sub Form_Load()
Positiv
End Sub

Private Sub Form_Click()
Positiv
End Sub
Private Sub Positiv()
Dim Valor As Integer
Dim Número As Single
Número = InputBox("Por favor,
introduzca un número", "Número")
Valor = Abs(CInt(Número))
Label1 = "El valor absoluto entero más
próximo es " & Valor
End Sub
```

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        Positiv()
    End Sub

Private Sub Form1_Click1(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Click
        Positiv()
End Sub

Private Sub Positiv()
   Dim Valor As Integer
   Dim Número As Single
Número = InputBox("Por favor, introduzca un número",
"Número")
Valor = Math.Abs(CInt(Número))
Label1.Text = "El valor absoluto entero más próximo es " &
Valor
End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

Las variables Valor y Número son variables locales al procedimiento Positiv, resultando no conocidas en el resto del programa.

EJERCICIO

Transformar en código el siguiente pseudocódigo referente a un programa denominado SUC01. El cometido del programa es calcular la suma de los términos de la sucesión:

$$\frac{1}{a}, \frac{1}{a-1}, \frac{1}{a-2}, \ldots, \frac{1}{1}$$

siendo *a* un número entero comprendido entre *1* y *100.*

```
PROGRAMA SUC 01 Curso Visual Basic aprenderaprogramar.com

    Variables
            Reales: a, Suma

            1. Inicio
                    2. Llamar EntraDatos
                    3. Llamar Calculo
                    4. Mostrar "El valor del sumatorio es", Suma
            5. Fin

            Módulo EntraDatos
                    1. Mientras a <= 0 ó a > 100 Hacer
                            1.1 Mostrar "Por favor introduzca un número entero
                                    comprendido entre 1 y 100"
                            1.2 Pedir a
                            1.3 a = Redondear(a)
                       Repetir
                    2. Mostrar "El dato base es", a
            FinMódulo

            Módulo Calculo
                    Variables
                            Enteras: i
                    1. Hacer
                            1.1 Suma = Suma + 1 / a
                            1.2 a = a – 1
                            1.3 i = i + 1
                       Repetir Mientras a <> 0
                    2. Mostrar "Contabilizados", i, "términos"
            FinMódulo
```

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN

Código versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit  '[Programa SUC01]
Dim a!, Suma As Single

Private Sub Form_Load()
EntraDatos
Cálculo
Label2 = "El valor del sumatorio es " &
Suma
End Sub

Private Sub EntraDatos()
Do While a <= 0 Or a > 100
   a = InputBox("Introduzca un entero
comprendido entre 1 y 100", "Entero")
   a = CInt(a)
Loop
Label1 = "El dato base es " & a
End Sub

Private Sub Cálculo()
Dim i As Integer
Do
   Suma = Suma + 1 / a
   a = a - 1
   i = i + 1
Loop While a <> 0
Label1 = Label1 & vbCrLf & vbCrLf &
"Contabilizados " & i & " términos"
End Sub
```

Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On '[Programa SUC01]
Public Class Form1
    Dim a, Suma As Single

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        EntraDatos()
        Cálculo()
        Label2.Text = "El valor del sumatorio es " &
Suma
    End Sub

    Private Sub EntraDatos()
        Do While a <= 0 Or a > 100
            a = InputBox("Introduzca un entero
comprendido entre 1 y 100", "Entero")
            a = CInt(a)
        Loop
        Label1.Text = "El dato base es " & a
    End Sub

    Private Sub Cálculo()
        Dim i As Integer
        Do
            Suma = Suma + 1 / a
            a = a - 1
            i = i + 1
        Loop While a <> 0
        Label1.Text = Label1.Text & vbCrLf & vbCrLf &
"Contabilizados " & i & " términos"
    End Sub
End Class
```

El programa trabaja con tres variables: a, Suma y i. Las variables a y Suma son de tipo Single, y por estar declaradas en la cabecera del módulo de formulario son conocidas en todos los procedimientos. La variable i es de tipo Integer y por estar declarada en la cabecera de un procedimiento es una variable local de dicho procedimiento. Si pruebas a escribir esta modificación: Label2 = "El valor del sumatorio es" & Suma & "con" & i & "términos", comprobarás que te sale en pantalla el mensaje: "Error de compilación: Variable no definida." En realidad la variable i sí está definida, pero tiene un ámbito restringido y no podemos usarla en el sitio donde hemos tratado de hacerlo.

Si ejecutas el programa se deben obtener resultados como:

a = 1 . El valor del sumatorio es 1.
a = 2. El valor del sumatorio es 1,5.
a = 3. El valor del sumatorio es 1,8333.
a = 5. El valor del sumatorio es 2,2833.
a = 10. El valor del sumatorio es 2,9289.
a = 50. El valor del sumatorio es 4,4992.

Próxima entrega: CU00340A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

MÓDULOS GENÉRICOS Y PARÁMETROS DE ENTRADA EN VISUAL BASIC.

Ya hemos visto que en Visual Basic una declaración de procedimiento siempre lleva unos paréntesis donde se pueden indicar parámetros de entrada. Los parámetros de entrada son indicados por el programador, o bien se generan automáticamente al tener lugar un evento y el propio ordenador envía ese parámetro de entrada al procedimiento que se dispara como consecuencia de que tenga lugar el evento. Veamos algunos ejemplos de procedimientos asociados a eventos.

Código 1 (versiones menos recientes VB)

```
Private Sub Text1_Change()

        .
        .
        .

End Sub
```

Código 1 (versiones más recientes VB)

```
Private Sub TextBox1_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox1.TextChanged

        .
        .
        .

End Sub
```

Código 2 (versiones menos recientes VB)

```
Private Sub Text1_KeyPress(KeyAscii
As Integer)

        .
        .
        .

End Sub
```

Código 2 (versiones más recientes VB)

```
Private Sub Text1_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles
TextBox1.KeyPress

        .
        .
        .

End Sub
```

El primer código corresponde a un procedimiento conducido por eventos que se invoca cuando el texto contenido en el TextBox Text1 cambia. En las versiones menos recientes de Visual Basic algunos procedimientos conducidos por eventos no tienen parámetros de entrada. En las versiones más recientes siempre suelen aparecer unos parámetros de entrada.

El segundo código sería el de un procedimiento conducido por eventos que se invoca cuando se produce la pulsación de una tecla en el TextBox1. En este caso, en las versiones menos recientes hay un parámetro (podría haber varios) que es enviado al procedimiento cuando tiene lugar el evento: un valor tipo Integer que se corresponde con el número asociado a la tecla pulsada según el código AscII. En las versiones más recientes esa información viene dentro del parámetro “e” que lleva distinta información asociada relacionada con el evento. Crea un TextBox, un Label y prueba el siguiente código:

© 2006-2029 aprenderaprogramar.com

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Private Sub Text1_KeyPress(KeyAscii As
Integer)
Label 1 = ""
Label 1 = KeyAscii
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub TextBox1_KeyPress(ByVal sender As Object,
ByVal e As System.Windows.Forms.KeyPressEventArgs)
Handles TextBox1.KeyPress
        Label1.Text = ""
        Label1.Text = "Valor numérico asociado a la tecla
pulsada es " & Microsoft.VisualBasic.Asc(e.KeyChar)
    End Sub
End Class
```

Comprobarás que cada vez que pulsas una tecla sobre el TextbBox aparece en pantalla un número: para la A, el 65, y para la a, el 97. Ese número es información que se pasa automáticamente al procedimiento cuando tiene lugar el evento. De esta forma se obtiene información útil sobre lo que está pasando (qué tecla ha pulsado el usuario, por dónde ha movido el ratón, qué botón del ratón ha pulsado, etc.) y se puede establecer un código previsto a ejecutar ante determinadas circunstancias.

Una declaración de procedimiento genérico que espera un parámetro de entrada podría ser esta:

Código versiones menos recientes VB:

```
Private Sub Raíz(Número As Single)

        .
        .
        .
End Sub
```

Código versiones más recientes VB:

```
Private Sub Raíz(ByVal Número As Single)
        .
        .
        .

End Sub
```

Dentro de los paréntesis se indican los parámetros de entrada requeridos, y el tipo de dato que han de ser (de momento no prestaremos atención a cuestiones adicionales como la aparición de la palabra ByVal). Si existen varios parámetros se escriben separados por comas, por ejemplo: (Número As Single, Valor As Double, Nombre As String). Si no se especifica tipo de dato para los parámetros serán del tipo por defecto (Variant en versiones menos recientes de Visual Basic u Object en las versiones más recientes). Nosotros siempre declararemos el tipo de los parámetros porque es una buena práctica de programación y no hacerlo podría dar lugar a diferentes problemas. La llamada al procedimiento podría hacerse con una de estas sintaxis:

```
Call Raíz(dato)
```

```
Raíz dato
```

```
Raíz  (dato)
```

La diferencia entre usar una u otra forma de llamada la veremos en breve.

© 2006-2029 aprenderaprogramar.com

Una declaración de función genérica que espera un parámetro de entrada podría ser esta:

Código versiones menos recientes VB:

```
Private Function Raíz(Número As Single) As Single

        .
        .
        .

End Sub
```

Código versiones más recientes VB:

```
Private Function Raíz(ByVal Número As Single) As Single

        .
        .
        .

End Function
```

Las llamadas a funciones se realizan escribiendo el nombre de la función seguido de los parámetros entre paréntesis. Dado que devuelven un resultado, se pueden usar para asignar contenido a una variable, para mostrar algo en pantalla, etc. Por ejemplo:

Resultado = Raíz(dato)
Label1 = Raíz(dato) ó Label1.Text = Raíz(dato)

Ejemplos de llamadas a un módulo genérico podrían ser estas:

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com

Option Explicit
Dim Dato As Single
Dim Resultado As Single

Private Sub Form_Load()
Call Raíz(9)
Call Raíz(-144)
Dato = InputBox("Introduzca número para
calcular raíz", "Dato")
Call Raíz(Dato)
End Sub

Private Sub Raíz(Número As Single)
Número = Abs(Número)
Resultado = Sqr(Número)
Label1 = Label1 & vbCrLf & "Raíz
calculada: " & Resultado
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

    Dim Dato As Single
    Dim Resultado As Single

Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Label1.Text = ""
        Call Raíz(9)
        Call Raíz(-144)
        Dato = InputBox("Introduzca número para
calcular raíz", "Dato")
        Call Raíz(Dato)
    End Sub

Private Sub Raíz(Número As Single)
        Número = Math.Abs(Número)
        Resultado = Math.Sqrt(Número)
        Label1.Text = Label1.Text & vbCrLf & "Raíz
calculada: " & Resultado
    End Sub
End Class
```

Si introducimos como número para calcular raíz 64 el programa nos mostrará: “Raíz calculada: 3; Raíz calculada: 12; Raíz calculada: 8”, donde 3 es la raíz de 9, 12 es la raíz de 144 (valor absoluto de -144), y 8 es la raíz de 64 (número que hemos introducido por teclado.

© 2006-2029 aprenderaprogramar.com

Si tratáramos de realizar una llamada del tipo Raíz("Nueva York") nos aparecería un mensaje de error como: “Error 13. No coinciden los tipos.” ó “La conversión de la cadena "Nueva york" en el tipo 'Single' no es válida.”

El mismo "objetivo" cumplido utilizando funciones lo exponemos a continuación. Recuerda que una función, además de ejecutar un código, devuelve un valor.

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
'Programa con function

Option Explicit
Dim Dato As Single
Dim Resultado As Single

Private Sub Form_Load()
Label1 = “”
Mostrar(Raíz(9))
Mostrar(Raíz(-144))
Dato = InputBox("Introduzca número
para calcular raíz", "Dato")
Mostrar(Raíz(Dato))
End Sub

Private Function Raíz(Número As Single)
As Single
Número = Abs(Número)
Raíz = Sqr(Número)
End Function

Private Sub Mostrar(valor As Single)
Label1 = Label1 & vbCrLf & "Raíz
calculada: " & valor
End Sub
```

Código versiones más recientes VB

```
REM Curso Visual Basic aprenderaprogramar.com
'Programa con function
Option Explicit On
Public Class Form1

    Dim Dato As Single
    Dim Resultado As Single

Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Label1.Text = ""
        Mostrar(Raíz(9))
        Mostrar(Raíz(-144))
        Dato = InputBox("Introduzca número para
calcular raíz", "Dato")
        Mostrar(Raíz(Dato))
End Sub

Private Function Raíz(ByVal Número As Single) As
Single
        Número = Math.Abs(Número)
        Raíz = Math.Sqrt(Número)
        Return Raíz
End Function

Private Sub Mostrar(ByVal valor As Single)
        Label1.Text = Label1.Text & vbCrLf & "Raíz
calculada: " & valor
End Sub
End Class
```

Próxima entrega: CU00341A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

PASO DE ARGUMENTOS POR REFERENCIA Y POR VALOR EN VISUAL BASIC.

Con Visual Basic se utilizan los términos "Paso de argumentos por Referencia (ByRef)", equivalente a lo que en el curso “Bases de la programación nivel II” de aprenderaprogramar.com hemos denominado transferencia de variable, y "Paso de argumentos por Valor (ByVal)", equivalente a lo que hemos denominado transferencia por valor.

Si no tienes claro el concepto de paso por referencia y paso por valor consulta el curso “Bases de la programación nivel II” de aprenderaprogramar.com.

El comportamiento de Visual Basic depende de la versión que estemos utilizando:

a) Con versiones menos recientes de Visual Basic: se permite no especificar cómo se pasa un parámetro. Por ejemplo Private Function suma (a As Integer, b As Integer). En caso de no estar especificado, la opción de defecto es que los parámetros pasan por referencia, con lo cual es posible que se modifique el valor de la variable inicial que se pasa como parámetro. En el último programa pasábamos la variable Dato como parámetro de la función Raíz. Al tener lugar la transformación del parámetro en valor absoluto, si introducíamos un número negativo su valor quedaba transformado. ¿Cómo hacer que la tranferencia sea por valor? Bastará con hacer la declaración de función o de procedimiento de la siguiente manera:

```
Private Function Raíz(ByVal Número As Single)
```

```
Private Sub Raíz(ByVal Número As Single)
```

Es decir, indicamos que el parámetro Número se procesa por valor.

b) Con versiones más recientes de Visual Basic: no se permite dejar sin especificar cómo se pasa un parámetro. Por ejemplo no se admite Private Function suma (a As Integer, b As Integer), de hecho si escribimos esto el editor lo transformará automáticamente en Private Function suma (ByVal a As Integer, ByVal b As Integer) ya que se asume que los parámetros pasarán por defecto por valor.

Para no tener dudas, lo más sencillo será especificar siempre cómo se pasa un parámetro a un procedimiento o una función. En caso de que no sepamos cómo pasarlo, recomendamos usar siempre ByVal para evitar que se generen modificaciones indeseadas dentro del procedimiento o función.

En la hipótesis de que existieran distintos parámetros y uno se quiera procesar de una manera y otro de otra, lo indicaremos antecediendo la palabra clave ByRef o ByVal delante del nombre del parámetro como en el siguiente ejemplo:

```
Private Function Raíz(ByVal Número As Single, ByRef x As Integer)
```

© 2006-2029 aprenderaprogramar.com

En las versiones menos recientes de Visual Basic, mientras que el modo de transferencia en las funciones sólo puede indicarse en la definición de la función, en los procedimientos podemos indicarlo bien en la declaración del procedimiento bien en la forma de llamada que empleemos. En versiones menos recientes de VB, el tipo de llamada condiciona cómo se transfiere el argumento. Así tenemos que:

a) Implican transferencia de variable llamadas del tipo: Call Raíz(dato) y Raíz dato
b) Implican transferencia de valor llamadas del tipo: Raíz(dato)

Se observa que es una diferencia mínima de escritura la que diferencia uno y otro tipo de llamada. Sin embargo, las consecuencias de usar uno u otro tipo pueden ser relevantes, y conocer y comprender estas implicaciones nos puede evitar más de un dolor de cabeza. Visual Basic no admite sintaxis del tipo Llamar Raíz(Dato) Por Valor.

Sin embargo, cuando en la declaración del procedimiento o función hemos indicado cómo se pasan los parámetros, resulta indistinto usar uno u otro tipo de sintaxis de llamada. Por tanto, en las versiones más recientes de Visual Basic, donde el modo de paso de parámetros es siempre indicado, no tiene relevancia cómo se haga la llamada.

EJERCICIO

Transformar en código el pseudocódigo que se muestra a continuación correspondiente a un programa denominado SUC02, cuyo objetivo es el cálculo del valor de un sumatorio del tipo:

$$\sqrt{a},\ \sqrt{a-1},\ \sqrt{a-2},\ \ldots,\ \sqrt{0}$$

```
[PROGRAMA SUC02 Curso Visual Basic aprenderaprogramar.com]

    Variables:
        Enteras: E
        Reales: Dato, Raiz01, Raiz02, Suce


    1. Inicio
        2. Mientras E <> 2 Hacer
            2.1 Mostrar "Elija  1. Cálculo  2. Salir"
            2.2 Pedir E
            2.3 Si E = 1 Entonces
                        Llamar EntraDatos
                        Llamar Proceso(Dato) PorValor
                        Llamar Resultados
                    FinSi
            Repetir
    3. Fin
```

© 2006-2029 aprenderaprogramar.com

```
Módulo EntraDatos
    1. Hacer
          Mostrar "Por favor, introduzca número entero entre 0 y 100"
          Pedir Dato
          Dato = Redondear(Dato)
       Repetir Mientras Dato < 0 ó Dato > 100
FinMódulo


Módulo Proceso(Num: Enteros)
    1. Raiz01 = SQR(Num)
    2. Raiz02 = – Raiz01
    3. Suce = 0
    4. Mientras Num >= 0 Hacer
          Suce = Suce + SQR(Num)
          Num = Num – 1
       Repetir
FinMódulo


Módulo Resultados
    Mostrar "Dato base =", Dato
    Mostrar "Raiz01=", Raiz01
    Mostrar "Raiz02=", Raiz02
    Mostrar "Valor de la suma de los términos de la sucesión =", Suce
FinMódulo
```

La solución se muestra en la siguiente página.

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN

Código versiones menos recientes VB:

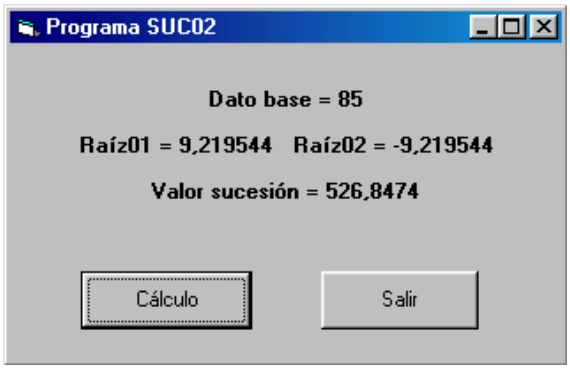
```
'Curso VB aprenderaprogramar.com
'Programa SUC02

Option Explicit
Dim Dato As Single
Dim Raíz01!, Raíz02 As Single
Dim Suce As Single

Private Sub Form_Load()
Form1.Caption = "Programa SUC02"
CommandCálculo.Caption = "Cálculo"
CommandSalir.Caption = "Salir"
End Sub

Private Sub CommandSalir_Click()
Unload Form1
End
End Sub

Private Sub CommandCálculo_Click()
Call EntraDatos
Call Proceso(Dato)
Call Resultados
End Sub

Private Sub EntraDatos()
Do
   Dato = CInt(InputBox("Por favor, introduzca
nº entero entre 0 y 100", "Dato"))
Loop While Dato < 0 Or Dato > 100
End Sub

Private Sub Proceso(ByVal Num As
Integer)
Raíz01 = Sqr(Num)
Raíz02 = -Raíz01
Suce = 0
Do While Num >= 0
   Suce = Suce + Sqr(Num)
   Num = Num - 1
Loop
End Sub

Private Sub Resultados()
Label1.Alignment = 2
Label1.FontBold = True
Label1 = "Dato base = " & Dato & vbCrLf &
vbCrLf & " Raíz01 = " & Raíz01 & " Raíz02 =" &
Raíz02 & vbCrLf & vbCrLf & "Valor sucesión = "
& Suce
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
'Programa SUC02

Option Explicit On
Public Class Form1
    Dim Dato As Single
    Dim Raíz01 As Single, Raíz02 As Single
    Dim Suce As Single

Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.Text = "Programa SUC02"
        ButtonCálculo.Text = "Cálculo"
        ButtonSalir.Text = "Salir"
        Label1.Text = ""
    End Sub

    Private Sub ButtonSalir_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ButtonSalir.Click
        Me.Close()
        End
    End Sub

    Private Sub ButtonCálculo_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ButtonCálculo.Click
        Call EntraDatos()
        Call Proceso(Dato)
        Call Resultados()
    End Sub

    Private Sub EntraDatos()
        Do
            Dato = CInt(InputBox("Por favor,
introduzca nº entero entre 0 y 100", "Dato"))
        Loop While Dato < 0 Or Dato > 100
    End Sub

    Private Sub Proceso(ByVal Num As Integer)
        Raíz01 = Math.Sqrt(Num)
        Raíz02 = -Raíz01
        Suce = 0
        Do While Num >= 0
            Suce = Suce + Math.Sqrt(Num)
            Num = Num - 1
        Loop
    End Sub

    Private Sub Resultados()
 Label1.TextAlign = ContentAlignment.MiddleCenter
 Label1.Font = New Font("Arial", 10, FontStyle.Bold)
 Label1.Text = "Dato base = " & Dato & vbCrLf &
vbCrLf & " Raíz01 = " & Raíz01 & " Raíz02 =" &
Raíz02 & vbCrLf & vbCrLf & "Valor sucesión = " &
Suce
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

El aspecto gráfico sería el siguiente:



En el curso “Bases de la programación nivel II” de aprenderaprogramar.com creamos una variante de pseudocódigo para este ejercicio consistente en el bloqueo del producto de los módulos. A modo de ejercicio adicional, te proponemos que crees esta variante en Visual Basic y compruebes las implicaciones que tiene.

Próxima entrega: CU00342A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

ARRAYS DINÁMICOS Y ARRAYS ESTÁTICOS. INSTRUCCIÓN REDIMENSIONAR (REDIM Y REDIM PRESERVE)

En el apartado correspondiente a declaración de variables ya hemos tratado la forma de declarar arrays estáticos unidimensionales y multidimensionales a través de la instrucción Dim. Usaremos esta misma instrucción para declarar arrays dinámicos. Recordemos que habíamos definido array dinámico como aquel que es declarado con un número de elementos componentes indefinido. La sintaxis a emplear será:

Dim [Nombre del array]() As [Tipo de datos]

Redim [Nombre del array]([dimensiones])

Es decir, en la cabecera de código del formulario o de un procedimiento haremos una declaración sin especificar dimensiones que luego concretaremos a través de la instrucción Redim. Por ejemplo:

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim A() As Integer

Private Sub Form_Load()
ReDim A(2)
A(1) = 3
Label1 = "El valor de A(1) es " & A(1)
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Dim A() As Integer
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        ReDim A(2)
        A(1) = 3
        Label1.Text = "El valor de A(1) es " & A(1)
End Sub
End Class
```

Dim A() As Integer supone la declaración del array dinámico (sin tamaño definido inicialmente).

ReDim A(2) establece que el array dinámico definido anteriormente tiene índice máximo 2.

En el caso de querer declarar una matriz (array o arreglo de varias dimensiones) como M (5, 7) que podría representar una tabla con datos como por ejemplo M (0,0) = 33, M (0,1) = 18, M (0,2) = 41 … etc. debemos hacerlo de la siguiente manera:

a) En las versiones menos recientes de Visual Basic declararemos inicialmente Dim M() As Integer y posteriormente ReDim M (5, 7). Una vez fijado que M tiene dos dimensiones ya no es posible cambiar el número de dimensiones por otro número.

b) En las versiones más recientes de Visual Basic declararemos inicialmente Dim M(,) As Integer, donde la coma indica que van a existir dos dimensiones que se declararán posteriormente. Si fuéramos a declarar posteriormente tres dimensiones escribiríamos Dim M (, ,) As Integer, para 4 dimensiones Dim M (, , ,) y así sucesivamente. No es posible cambiar el número de dimensiones de un array dinámico, que será el declarado inicialmente.

© 2006-2029 aprenderaprogramar.com

Un array estático no puede ser redimensionado. En cambio, uno dinámico puede sufrir múltiples redimensionamientos. No se puede declarar el tipo de dato cada vez que redimensionemos. Hay que hacerlo al declarar la matriz dinámica, una sola vez.

Por defecto, el uso de Redim implicará perder todos los datos anteriormente contenidos en la matriz (array, arreglo) que se redimensiona. Para conservar los datos anteriores podemos hacerlo de dos maneras. Por un lado, usando la cláusula Preserve según esta sintaxis:

```
Redim Preserve [Nombre del array]([dimensiones])
```

La limitación de usar Preserve viene dada porque únicamente se nos permite cambiar la última dimensión del array (la situada más a la derecha). Tampoco se permite cambiar el número de dimensiones. Si estamos trabajando con un array unidimensional no habrá problemas, pero con otro tipo de arrays puede haberlos.

Ejemplo: si declaramos un array multidimensional denominado B como Dim B(, ,) As Integer (en versiones menos recientes simplemente Dim B () As Integer), luego redimensionamos ReDim B(4, 2, 6) y posteriormente intentamos hacer ReDim B (2,2,2) obtendremos un mensaje de error ya que sólo puede cambiarse la dimensión más a la derecha. Es decir, ReDim (4, 2, x) donde x es la dimensión cuyo tamaño cambiamos es válido. Pero cambiar las dimensiones anteriores no es válido.

Cuando se utiliza Preserve aumentando el número de elementos del array, los nuevos elementos tienen valor cero o nulo y los que existían anteriormente conservan su valor. Cuando se trata de una disminución de los elementos de un array, aquellos contenidos en índices que desaparecen dejan de existir a todos los efectos, mientras que los existentes en dimensiones que se conservan mantienen su valor. En caso de arrays de varias dimensiones, con Redim Preserve sólo podemos cambiar la última dimensión, al igual que hemos visto antes. Para un array dinámico que se ha redimensionado como A(5, 5, 5):

a) Es válido: Redim Preserve A(5, 5, 8)
   Modifica la última dimensión.

b) No es válido: Redim Preserve A(5, 8, 5)
   Modifica una dimensión que no es la última.

Para conservar el contenido de arrays cuando no es posible a través de Preserve habremos de hacerlo declarando otro array con las dimensiones deseadas y transfiriendo elemento a elemento los valores desde un array a otro.

Una cuestión a tener en cuenta es que la instrucción Erase, cuyo funcionamiento habíamos visto que con matrices estáticas daba lugar a que todos los elementos de la matriz se convirtieran en ceros o cadenas vacías, al ser aplicada a un array dinámico da lugar a que este desaparezca totalmente. Por tanto, no podremos invocar la matriz después de usar Erase a no ser que la declaremos de nuevo usando ReDim nuevamente. Para borrar el contenido de una matriz dinámica bastará con usar la instrucción Redim sin variar las dimensiones, o bien establecer el valor de cada uno de los elementos a cero o cadena vacía a través del código.

© 2006-2029 aprenderaprogramar.com

En las versiones menos recientes de Visual Basic era posible variar el número de localizadores de un array con ReDim (sin Preserve), tanto para aumentarlos como para disminuirlos, usando código de este tipo (siempre sin Preserve):

```
'Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A() As Integer

Private Sub Form_Load()
ReDim A(2, 3, 4, 5)
A(1, 1, 1, 1) = 3
Label1 = "El valor de A(1,1,1,1) es " & A(1, 1, 1, 1)
ReDim A(2)  '[Reducción del número de localizadores]
End Sub
```

Donde observamos que el array *A*, declarado inicialmente como de cuatro dimensiones, es redimensionado para dejarlo como unidimensional. En las versiones más recientes de Visual Basic esto no es posible, porque la declaración del array A tiene que establecer un número de dimensiones fijas para el mismo. En este ejemplo tendríamos que declarar Dim A(, , ,) As Integer y de este modo el número de dimensiones de A será siempre 4.

EJERCICIO

Crear el código correspondiente al siguiente pseudocódigo de un programa MATRIZ01 como ejemplo del redimensionamiento de arrays. Se pretende declarar un array de dimensiones (3,3) donde se usan los índices 1 a 3 y cada elemento tiene como contenido el valor de la suma de sus índices. Por ejemplo Matriz (1,1) = 2, Matriz (1,2) = 3, Matriz (1,3) = 4, Matriz (2,1) = 3, Matriz (2,2) = 4, etc. Una vez creado, se quiere reducir el tamaño de la matriz a (2,2) conservando en ella los valores que inicialmente existían en los índices que no se han eliminado y posteriormente volver a redimensionar a dimensiones (3,3) y mostrar el contenido de todos los elementos de la matriz.

```
[Curso VB aprenderaprogramar.com]

    Variables
        Enteras: Matriz(), i, j

    1. Inicio
        2. Redimensionar Matriz(3,3)
        3. Desde i = 1 hasta 3
            3.1 Desde j = 1 hasta 3
                3.1.1 Matriz(i,j) = i + j
            Siguiente
        Siguiente
        4. Redimensionar Matriz(2,2) conservando elementos
        5. Mostrar "Se han eliminado 9 – 4 = 5 elementos"
        6. Redimensionar Matriz(3,3) conservando elementos
        6. Mostrar todos los elementos de la matriz
    8. Fin
```

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim Matriz() As Integer
Dim Transitoria() As Integer
Dim i%, j As Integer

Private Sub Form_Load()
ReDim Matriz(3, 3)
Label1 = ""
For i = 1 To 3
   For j = 1 To 3
      Matriz(i, j) = i + j
   Next j
Next i
ReDim Transitoria(2, 2)
For i = 1 To 2
   For j = 1 To 2
      Transitoria(i, j) = Matriz(i, j)
   Next j
Next i
ReDim Matriz(2, 2)
Label1 = Label1 & "Se han eliminado 9 - 4 = 5
elementos" & vbCrLf & vbCrLf
ReDim Matriz(3, 3)
For i = 1 To 2
   For j = 1 To 2
      Matriz(i, j) = Transitoria(i, j)
   Next j
Next i
Label1 = Label1 & "La variable Matriz(3,2) vale "
& Matriz(3, 2) & vbCrLf & vbCrLf
For i = 1 To 3
   For j = 1 To 3
      Label1 = Label1 & "Matriz (" & i & "," & j &
") =" & Matriz(i, j) & "    "
   Next j
Next i
End Sub
```

Código versiones más recientes VB

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1

Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Dim Matriz(,) As Integer
        Dim Transitoria(,) As Integer
        Dim i, j As Integer

        ReDim Matriz(3, 3)
        Label1.Text = ""
        For i = 1 To 3
            For j = 1 To 3
                Matriz(i, j) = i + j
            Next j
        Next i
        ReDim Transitoria(2, 2)
        For i = 1 To 2
            For j = 1 To 2
                Transitoria(i, j) = Matriz(i, j)
            Next j
        Next i
        ReDim Matriz(2, 2)
        Label1.Text = Label1.Text & "Se han
eliminado 9 - 4 = 5 elementos" & vbCrLf & vbCrLf
        ReDim Matriz(3, 3)
        For i = 1 To 2
            For j = 1 To 2
                Matriz(i, j) = Transitoria(i, j)
            Next j
        Next i
        Label1.Text = Label1.Text & "La variable
Matriz(3,2) vale " & Matriz(3, 2) & vbCrLf &
vbCrLf
        For i = 1 To 3
            For j = 1 To 3
                Label1.Text = Label1.Text &
"Matriz (" & i & "," & j & ") =" & Matriz(i, j) &
"    "
            Next j
        Next i
    End Sub
End Class
```

Comentarios: Dado que pretendemos modificar varias dimensiones de la matriz, no podemos utilizar Preserve y hacerlo automáticamente. Por ello creamos una matriz transitoria que utilizamos para evitar que se pierdan los valores que nos interesan.

El resultado es el que esperábamos con el pseudocódigo pero el tener que usar matrices transitorias complica un poco la programación.

© 2006-2029 aprenderaprogramar.com

Tras ejecutar el programa obtendremos este resultado:

Se han eliminado 9 - 4 = 5 elementos

La variable Matriz (3,2) vale 0

Matriz (1,1) =2 Matriz (1,2) =3 Matriz (1,3) =0 Matriz (2,1) =3 Matriz (2,2) =4 Matriz (2,3) =0
Matriz (3,1) =0 Matriz (3,2) =0 Matriz (3,3) =0

Podemos comprobar que todos los elementos de índice 3 tienen valor cero.

Próxima entrega: CU00343A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

MANEJO DE DATOS CON ARRAYS. LÍMITES SUPERIOR (UBOUND) E INFERIOR (LBOUND).

Definido un array A(n) donde n define el índice más grande que puede tener un elemento del array, la función Ubound(A) nos devuelve el valor de n y el valor LBound(A) nos devuelve el valor del índice más bajo que puede tener el array (que normalmente será 0).

En las versiones menos recientes de Visual Basic se permitían declaraciones del tipo A(m To n), estando los índices del array comprendidos entre m y n. En este caso la función Lbound(A) nos devuelve el valor del índice menor del array (m) y la instrucción Ubound(A) devuelve el valor del índice mayor (n).

En general el índice menor por defecto en un array será 0. No obstante, en las versiones menos recientes de Visual Basic se permitía establecer como primer índice de un array por defecto el 1 mediante la instrucción Option Base escribiendo Option Base 1. Si se hace uso de esa opción los resultados de Lbound se pueden ver afectados por el valor establecido para Option Base, obteniéndose un 0 o un 1 según el caso. En las versiones más recientes de Visual Basic el índice inferior será siempre el cero.

El número de elementos del array será m – n + 1, donde m es el límite superior y n el límite inferior. Por ejemplo un array declarado como Dim A(3) As Integer consta de 3-0+1 = 4 elementos que son A(0), A(1), A(2) y A(3). Prueba el siguiente código:

Código versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim A(45)

Private Sub Form_Load()
Label1 = "Indice menor del array A = "
& LBound(A) & vbCrLf & vbCrLf
Label1 = Label1 & "Indice mayor del
array A = " & UBound(A) & vbCrLf &
vbCrLf
Label1 = Label1 & "Array con " &
UBound(A) - LBound(A) + 1 & "
elementos"
End Sub
```

Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim A(45)

Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

        Label1.Text = "Indice menor del array A = " &
LBound(A) & vbCrLf & vbCrLf
        Label1.Text = Label1.Text & "Indice mayor del
array A = " & UBound(A) & vbCrLf & vbCrLf
        Label1.Text = Label1.Text & "Array con " &
UBound(A) - LBound(A) + 1 & " elementos"
    End Sub
End Class
```

En el caso de arrays multidimensionales se hace necesario indicar si queremos referirnos al primer localizador, al segundo, tercero, etc. Así Lbound(A, 3) nos devolvería el índice menor del array A en su dimensión 3. Por ejemplo:

Para: Dim A(55, 20, 33, 60)

LBound(A, 1) devuelve 0, UBound(A, 1) devuelve 55, LBound(A, 2) devuelve 0, UBound(A, 2) devuelve 20, LBound(A, 3) devuelve 0, UBound(A, 3) devuelve 33, LBound(A, 4) devuelve 0, UBound(A, 4) devuelve 60.

© 2006-2029 aprenderaprogramar.com

EJERCICIO

Desarrollar el pseudocódigo y diagrama de flujo para un algoritmo que calcule la superficie de un terreno que le corresponde a un heredero después de n generaciones, partiendo de una superficie inicial en la generación cero. Se supone que hay división a partes iguales entre herederos

SOLUCIÓN

Código versiones menos recientes de VB:

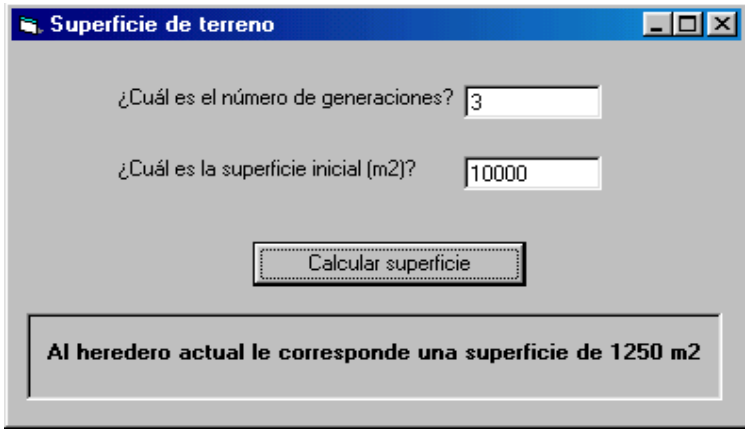
```
'Curso VB aprenderaprogramar.com
'[Superficie de terreno]

Option Explicit
Dim i%, n As Integer
Dim Supin!, Toca As Single
Dim Hgen() As Integer

Private Sub Form_Load()
Text1 = "": Text2 = ""
Form1.Caption = "Superficie de terreno"
CommandCalcular.Caption = "Calcular
superficie"
Label1.Caption = "¿Cuál es el número de
generaciones?"
Label2.Caption = "¿Cuál es la superficie inicial
(m2)?"
End Sub

Private Sub CommandCalcular_Click()
n = Val(Text1)
ReDim Hgen(n)
Supin = Val(Text2)
Toca = Supin
For i = 1 To n
   Hgen(i) = InputBox("¿Cuál es el número de
herederos de la generación " & i & "?", "Nº
herederos")
   Toca = Toca / Hgen(i)
Next i
LabelFinal.Alignment = 2
LabelFinal.FontBold = True
LabelFinal = vbCrLf & "Al heredero actual le
corresponde una superficie de " & Toca & " m2"
End Sub
```

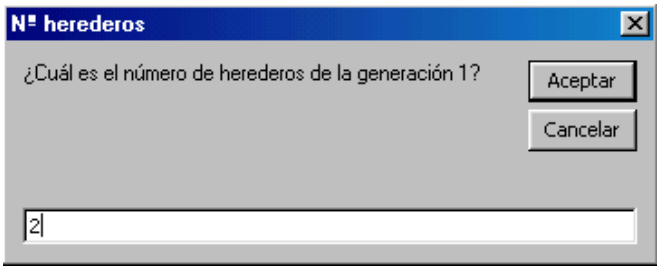
Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        TextBox1.Text = "" : TextBox2.Text = ""
        Me.Text = "Superficie de terreno"
        ButtonCalcular.Text = "Calcular
superficie"
        Label1.Text = "¿Cuál es el número de
generaciones?"
        Label2.Text = "¿Cuál es la superficie
inicial (m2)?"
    End Sub

    Private Sub ButtonCalcular_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles ButtonCalcular.Click
        Dim i, n As Integer
        Dim Supin, Toca As Single
        Dim Hgen() As Integer

        n = Val(TextBox1.Text)
        ReDim Hgen(n)
        Supin = Val(TextBox2.Text)
        Toca = Supin
        For i = 1 To n
        Hgen(i) = InputBox("¿Cuál es el número de
herederos de la generación " & i & "?", "Nº
herederos")
            Toca = Toca / Hgen(i)
        Next i
LabelFinal.Font = New Font("Arial", 12,
FontStyle.Bold)
LabelFinal.TextAlign = ContentAlignment.MiddleCenter
        LabelFinal.text = vbCrLf & "Al heredero
actual le corresponde una superficie de " & Toca &
" m2"
    End Sub
End Class
```

Comentarios: Para la generación 1, 2, 3,... el número de herederos correspondiente se almacena en el vector Hgen(i), que se declara como array dinámico por no ser conocido el dato de número de herederos a priori. Gráficamente el programa sería este:

© 2006-2029 aprenderaprogramar.com





A modo de ejemplo, si los datos de entrada son 3 generaciones, superficie inicial 10000 m2, y en cada generación hay dos herederos, obtenemos como resultado que al heredero actual le corresponde una superficie de 1250 m2.

EJERCICIO

Nos han facilitado el siguiente pseudocódigo y una explicación relativa al objetivo de un programa que debemos realizar:

```
1. Inicio [Extracción de datos p100 curso VB aprenderaprogramar.com]
        2. n = 1
        3. Mientras Esperado = Falso Hacer
                3.1 Desde i = n hasta n + 99 Hacer
                        Leer Dato(i)
                        Si Dato(i) > 600 y Dato(i) < 700 Entonces
                                j = j + 1
                        FinSi
                    Siguiente
                3.2 Si j > 100 Entonces
                        Esperado = Verdadero
                    FinSi
                3.3 Si Esperado = Verdadero Entonces
                        Mostrar "Se cumple lo previsto habiendo
                        cumplido", j, "datos de un total de", n + 99,
                        "datos extraídos"
                     SiNo
                        Mostrar "Extracción de un nuevo paquete"
                        n = n + 100
                     FinSi
            Repetir
4. Fin
```

© 2006-2029 aprenderaprogramar.com

Nos indican que se trata de extraer datos de un archivo existente en “paquetes” de 100 unidades. Si el número de datos extraídos con valor superior a 600 y menor de 700 es mayor de 100, se debe mostrar el mensaje “Se cumple lo previsto”, indicando el número de datos que se extrajeron cumpliendo los requisitos frente al total extraído. En caso contrario, continuar extrayendo paquetes de datos (se consideran ilimitados). Considerar que los datos se encuentran en un archivo creado previamente.

SOLUCIÓN

Para generar una simulación del archivo con los datos vamos a usar el siguiente código (atención a poner una ruta de archivo correcta, donde queramos guardar el archivo):

Código versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim Canal As Integer
Dim i As Integer

Private Sub Form_Load()
Canal = FreeFile
Open "C:\misdatos.dat" For Output As Canal
For i = 1 To 300
Randomize
Write #Canal, Int((750 - 550 + 1) * Rnd + 550)
Next i
Close
Label1 = “Datos generados correctamente”
End Sub
```

Código versiones más recientes de VB

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
Dim myFileToWrite As New
System.IO.StreamWriter("C:\Users\Asus\Desktop\misdat
osnet.dat", False)
        For i = 1 To 300
          Randomize()
          myFileToWrite.WriteLine(Int((750 - 550 +
1) * Rnd() + 550))
        Next i
        myFileToWrite.Close()
    Label1.Text = "Datos generados correctamente"
    End Sub
End Class
```

Hemos establecido un rango de datos entre 550 y 750 para provocar que los datos estén en el entorno de los valores esperados. En caso contrario se puede producir que no exista un número suficiente de datos válidos y se genere un error al tratar de extraer más datos de los realmente existentes en el archivo. El programa se muestra a continuación:

© 2006-2029 aprenderaprogramar.com

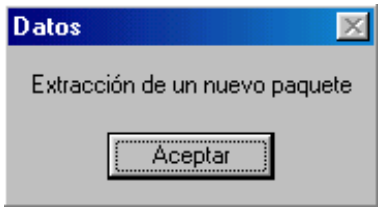
Código versiones menos recientes de VB:

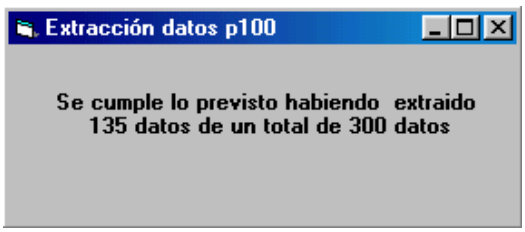
```
'Curso VB aprenderaprogramar.com
'[Extracción datos p100]
Option Explicit
Dim Canal As Integer
Dim i%, j%, n As Integer
Dim Esperado As Boolean
Dim Dato() As Integer

Private Sub Form_Load()
Form1.Caption = "Extracción datos p100"
LabelResultado.Alignment = 2
LabelResultado.FontBold = True
n = 1
Canal = FreeFile
Open "C:\misdatos.dat" For Input As Canal
Do While Esperado = False
   ReDim Dato(n + 99)
   For i = n To n + 99
      Input #Canal, Dato(i)
      If Dato(i) > 600 And Dato(i) < 700 Then
        j = j + 1
      End If
   Next i
   If j > 100 Then
      Esperado = True
   End If
   If Esperado = True Then
      LabelResultado = "Se cumple lo previsto
habiendo cumplido " & j & " datos de un total
de " & n + 99 & " datos extraidos"
   Else
   MsgBox("Extracción de un nuevo paquete")
   n = n + 100
   End If
Loop
Close
End Sub
```

Código versiones más recientes de VB

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim i, j, n As Integer
        Dim Esperado As Boolean
        Dim Dato() As Integer

        Me.Text = "Extracción datos p100"
        LabelResultado.Font = New Font("Arial", 12,
FontStyle.Bold)
        LabelResultado.TextAlign =
ContentAlignment.MiddleCenter
        n = 1
        Dim myFileToRead As New
System.IO.StreamReader("C:\Users\Asus\Desktop\misdat
osnet.dat", False)
        Do While Esperado = False
            ReDim Dato(n + 99)
            For i = n To n + 99
                Dato(i) = myFileToRead.ReadLine()
            If Dato(i) > 600 And Dato(i) < 700 Then
                    j = j + 1
            End If
            Next i
            If j > 100 Then
                Esperado = True
            End If
            If Esperado = True Then
                LabelResultado.Text = "Se cumple lo
previsto habiendo cumplido " & j & " datos de un
total de " & n + 99 & " datos extraidos"
            Else
            MsgBox("Extracción de un nuevo paquete")
            n = n + 100
            End If
        Loop
        myFileToRead.Close()
    End Sub
End Class
```

Gráficamente:





Próxima entrega: CU00344A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

EJERCICIO RESUELTO

Generar el código correspondiente al siguiente ejercicio, relativo al cálculo de un valor medio a partir de una serie de datos que terminan en un señalero (centinela).

En un fichero tenemos las notas de un grupo de alumnos de una clase en un array Nota(1), Nota(2), ..., Nota(n – 1), establecidas entre 0 y 10. El último dato es un señalero con valor – 50. Se desea crear un programa que determine la nota media usando un array para almacenar las notas extraídas del fichero que inicialmente tenga índice máximo 10 y se amplíe a 20 en caso de que llegue a completarse el número de elementos admitidos en el array (y en caso de llegar al índice 20 se amplíe a 30 y así sucesivamente.

SOLUCIÓN

Consideraremos que el fichero notas.dat contiene los siguientes datos que nos servirán para probar el programa: 5, 7, 2, 3, 9, 5, 6, 8, - 50 (cada dato estará en una línea distinta, aquí los ponemos separados por comas por motivo de espacio).

Usaremos un label denominado Labelfinal para mostrar los resultados.

Código versiones VB menos recientes:

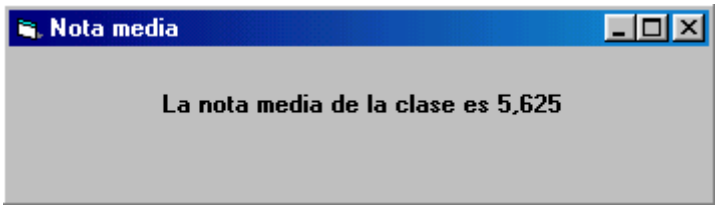
```
'[Curso VB aprenderaprogramar.com]
'[Determinación de media]

Option Explicit
Dim Canal As Integer
Dim i%, Suma As Single
Dim Nota() As Single
Dim Media As Single

Private Sub Form_Load()
Form1.Caption = "Nota media"
Labelfinal.Alignment = 2
Labelfinal.FontBold = True
ReDim Nota(10)
Canal = FreeFile
Open "C:\notas.dat" For Input As Canal
i = 1
Input #Canal, Nota(i)
Do While Nota(i) >= 0
Suma = Suma + Nota(i)
If UBound(Nota) = i Then ReDim Preserve
Nota(i + 10)
i = i + 1
Input #Canal, Nota(i)
Loop
Media = Suma / (i - 1)
Close
Labelfinal = "La nota media de la clase es " &
Media
End Sub
```

Código versiones VB más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1

Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim i As Integer
        Dim Suma As Single
        Dim Nota() As Single
        Dim Media As Single

        Me.Text = "Nota media"
Labelfinal.Font = New Font("Arial", 12,
FontStyle.Bold)
Labelfinal.TextAlign = ContentAlignment.MiddleCenter
ReDim Nota(10)
Dim myFileToRead As New System.IO.StreamReader
("C:\Users\Asus\Desktop\notas2.dat", False)
        i = 1
        Nota(i) = myFileToRead.ReadLine()

        Do While Nota(i) >= 0
        Suma = Suma + Nota(i)
        If UBound(Nota) = i Then ReDim Preserve
Nota(i + 10)
        i = i + 1
        Nota(i) = myFileToRead.ReadLine()
        Loop
        Media = Suma / (i - 1)
        myFileToRead.Close()
        Labelfinal.Text = "La nota media de la
clase es " & Media
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

El aspecto gráfico resultado al ejecutar el programa será similar a este:



Comentarios: En este caso, se establece una extracción de paquetes de datos de 10 en 10. Fijémonos en una línea: If UBound(Nota) = i Then ReDim Preserve Nota(i + 10)

Aquí establecemos que si el número de elementos que se extraen coincide con el tamaño máximo del vector nota que está recibiendo los datos del archivo, dicho vector se amplía en otros diez elementos. Cuando se llega al señalero de terminación es posible que cierto número de elementos del array, entre uno y diez, queden vacíos. Cuanto más grande hagamos el intervalo asociado a cada redimensionamiento, menor número de procesos es necesario. En cambio, mayor cantidad de elementos del array corremos el riesgo de que queden vacíos.

EJERCICIO RESUELTO

Generar el código para un programa que pide un número impar y realiza cálculos numéricos. En concreto el programa presentará las siguientes opciones:

a) Introducir un valor entero impar comprendido entre 1 y 19
b) Calcular la serie numérica 1 + 3 + 5 + ··· + n
c) Calcular 1 * 3 * 5 * ··· * n
d) Salir del programa.

Nota: Los cálculos se han de realizar a través de un vector que almacene los valores impares comprendidos entre 1 y n.

El pseudocódigo es el siguiente:

```
PROGRAMA IMPARES01 Curso Visual Basic aprenderaprogramar.com

    Variables
        Enteras: E, Dato, Cantidad, Valor()

1.  Inicio
    2.  Mientras E <> 4 Hacer
        2.1  Si E = 1 Entonces
                 Llamar Genera
             FinSi
        2.2  Si E = 2 y Dato <> 0 Entonces
                 Llamar CalculaSuma(Cantidad) PorValor
             FinSi
```

© 2006-2029 aprenderaprogramar.com

```
                2.3  Si E = 3 y Dato <> 0 Entonces
                        Llamar CalculaMult(Cantidad) PorValor
                     FinSi
                2.4  Mostrar "Elija opción:  1. Introducir dato
                                             2. Calcular 1 + 3 + 5 + ··· + n
                                             3. Calcular 1 * 3 * 5 * ··· * n
                                             4. Salir"
Pedir E
            Repetir
    3.  Fin

    Módulo Genera
        Variables
            Enteras: Impar, i
            Booleanas: Par
        1.  Dato = 0
            2.  Mientras Dato <= 0 ó Dato > 19 ó Par = Verdadero Hacer
                    Par = Falso
                    Mostrar "Introduzca un número impar entre 1 y 19"
                    Pedir Dato
                    Si Dato mod 2 = 0 Entonces
                            Par = Verdadero : Mostrar "El número debe ser impar"
                    FinSi
                Repetir
            3.  Impar = – 1 : i = 0
            4.  Hacer
                    i = i + 1
                    Impar = Impar + 2
                    Redimensionar Valor(i)
                    Valor(i) = Impar
                Repetir Mientras Impar < Dato
            5.  Cantidad = i
    FinMódulo

    Módulo CalculaSuma(Numero: Enteros)
        Variables
            Enteras: i, Suma
        1.  Suma = 0
            2.  Desde i = 1 hasta Numero Hacer
                    Suma = Suma + Valor(i)
                Siguiente
            3.  Mostrar "La suma 1 + 3 + 5 + ··· + n vale", Suma
    FinMódulo

    Módulo CalculaMult(Cifra: Enteros)
        Variables
            Enteras: i, Acumulado
        1.  Acumulado = 1
            2.  Desde i = 1 hasta Cifra Hacer
                    Acumulado = Acumulado * Valor(i)
                Siguiente
            3.  Mostrar "El producto 1 * 3 * 5 * ··· * n vale", Acumulado
    FinMódulo
```

SOLUCIÓN

Vamos a plantear un programa donde inicialmente se nos mostrarán cuatro botones: uno que llevará por título "Introducir valor" y que permitirá establecer el valor de n, otro denominado "Calcular serie 1+3+5…" que hará que se ejecute el cálculo con el valor de n existente, otro denominado "Calcular serie 1*3*5…" con igual función para obtener el valor de las multiplicaciones, y finalmente un botón "Salir" que permitirá salir del programa.

© 2006-2029 aprenderaprogramar.com

Código versiones VB menos recientes:

```
'[Curso VB aprenderaprogramar.com]
'[PROGRAMA IMPARES01]

Option Explicit
Dim Dato%, Cantidad%, Valor() As Integer

Private Sub Form_Load()
Form1.Caption = "Impares01"
LabelResultado.Alignment = 2
LabelResultado.FontBold = True
Command1.Caption = "Introducir valor"
Command2.Caption = "Calcular serie
1+3+5..."
Command3.Caption = "Calcular serie 1*3*5..."
Command4.Caption = "Salir"
End Sub

Private Sub Command1_Click()
Call Genera
End Sub

Private Sub Command2_Click()
If Dato <> 0 Then Call
CalculaSuma(Cantidad)
End Sub

Private Sub Command3_Click()
If Dato <> 0 Then Call CalculaMult(Cantidad)
End Sub

Private Sub Command4_Click()
Unload Form1
End
End Sub

Private Sub Genera()
Dim Impar%, i As Integer
Dim Par As Boolean
Dato = 0
Do While Dato <= 0 Or Dato > 19 Or Par =
True
   Par = False
   Dato = Val(InputBox("Introduzca un número
impar entre 1 y 19", "Impar"))

   If Dato Mod 2 = 0 Then
      Par = True
      MsgBox("El número debe ser impar")
   End If
Loop

'Continúa…
```

Código versiones VB más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
'[PROGRAMA IMPARES01]

Option Explicit On
Public Class Form1
    Dim Dato, Cantidad, Valor() As Integer

Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Me.Text = "Impares01"
LabelResultado.Font = New Font("Arial", 12,
FontStyle.Bold)
LabelResultado.TextAlign =
ContentAlignment.MiddleCenter
        Button1.Text = "Introducir valor"
        Button2.Text = "Calcular serie 1+3+5..."
        Button3.Text = "Calcular serie 1*3*5..."
        Button4.Text = "Salir"
    End Sub

Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Call Genera()
End Sub

Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
       If Dato <> 0 Then Call
CalculaSuma(Cantidad)
End Sub

Private Sub Button3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button3.Click
       If Dato <> 0 Then Call
CalculaMult(Cantidad)
End Sub

Private Sub Button4_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button4.Click
        Me.Close()
        End
End Sub

Private Sub Genera()
     Dim Impar, i As Integer
     Dim Par As Boolean
     Dato = 0
     Do While Dato <= 0 Or Dato > 19 Or Par = True
            Par = False
            Dato = Val(InputBox("Introduzca un
número impar entre 1 y 19", "Impar"))

            If Dato Mod 2 = 0 Then
                Par = True
                MsgBox("El número debe ser impar")
            End If
     Loop

'Continúa…
```

© 2006-2029 aprenderaprogramar.com

```
'Continuación del código
Impar = -1: i = 0
Do
   i = i + 1
   Impar = Impar + 2
   ReDim Preserve Valor(i)
   Valor(i) = Impar
Loop While Impar < Dato
Cantidad = i
LabelResultado = " "
End Sub

Private Sub CalculaSuma(ByVal Número As Integer)
Dim i%, Suma As Integer
Suma = 0
For i = 1 To Número
   Suma = Suma + Valor(i)
Next i
LabelResultado = vbCrLf & "La suma
1+3+5+...+n vale " & Suma
End Sub

Private Sub CalculaMult(ByVal Cifra As Integer)
Dim i As Integer
Dim Acumulado As Single
Acumulado = 1
For i = 1 To Cifra
   Acumulado = Acumulado * Valor(i)
Next i
LabelResultado = vbCrLf & "El producto
1*3*5*...*n vale " & Acumulado
End Sub
```

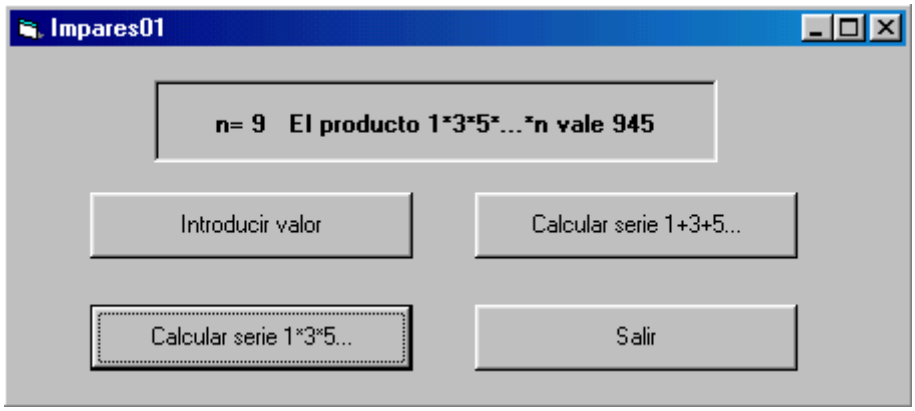
```
'Continuación del código
        Impar = -1 : i = 0
        Do
            i = i + 1
            Impar = Impar + 2
            ReDim Preserve Valor(i)
            Valor(i) = Impar
        Loop While Impar < Dato
        Cantidad = i
        LabelResultado.Text = " "
    End Sub


Private Sub CalculaSuma(ByVal Número As Integer)
        Dim i, Suma As Integer
        Suma = 0
        For i = 1 To Número
            Suma = Suma + Valor(i)
        Next i
        LabelResultado.Text = vbCrLf & "La suma
1+3+5+...+n vale " & Suma
End Sub


Private Sub CalculaMult(ByVal Cifra As Integer)
        Dim i As Integer
        Dim Acumulado As Single
        Acumulado = 1
        For i = 1 To Cifra
            Acumulado = Acumulado * Valor(i)
        Next i
        LabelResultado.Text = vbCrLf & "El
producto 1*3*5*...*n vale " & Acumulado
End Sub
End Class
```

Comentarios: Distintas variantes pueden ser introducidas respecto a lo planteado en pseudocódigo. Sin embargo, hemos tratado de respetar en la medida de lo posible el esquema allí propuesto. La única variación introducida es la definición de la variable Acumulado como tipo real (Single) en vez de entera (Integer). A ello estamos obligados ya que el rango de valores admisible para un tipo Integer con Visual Basic puede resultar insuficiente para los rangos numéricos con que se hace trabajar a esta variable, que pueden ser muy grandes. Obviar este hecho puede dar lugar a que a partir de un valor introducido igual o superior a cierto número se nos presente un mensaje de "Error 6 en tiempo de ejecución: desbordamiento". Para solventar este problema definimos la variable como tipo Single.

Hemos usado Redim Preserve en lugar de Redim para conservar los valores anteriores del vector Valor. Dicho vector se redimensiona en cada pasada del bucle, aunque podíamos haber optado por redimensionarlo cada cierto número de pasadas o asignarle un valor grande inicialmente y reducirlo al valor definitivo después. Estos y otros aspectos del programa se comentan en el curso “Bases de la programación nivel I” de aprenderaprogramar.com, donde se explica la lógica para la creación del programa y el diagrama de flujo del mismo.

© 2006-2029 aprenderaprogramar.com

El aspecto gráfico será algo así:



Próxima entrega: CU00345A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

EJERCICIO RESUELTO VISUAL BASIC

Crear el código correspondiente al siguiente ejercicio (mismo ejercicio que hemos resuelto anteriormente) pero, en este caso, prescindiendo del uso de arrays.

Se trata de crear un programa con las siguientes opciones:

a) Introducir un valor entero impar comprendido entre 1 y 19

b) Calcular la serie numérica 1 + 3 + 5 + ··· + n

c) Calcular 1 * 3 * 5 * ··· * n

d) Salir del programa.

SOLUCIÓN

El formato gráfico será el mismo que para el programa anterior, aunque en este caso no hemos usado arrays.

Código versiones VB menos recientes:

```
'[Curso VB aprenderaprogramar.com]
'[PROGRAMA IMPARES02]

Option Explicit
Dim Dato As Integer

Private Sub Form_Load()
Form1.Caption = "Impares02"
LabelResultado.Alignment = 2
LabelResultado.FontBold = True
Command1.Caption = "Introducir valor"
Command2.Caption = "Calcular serie
1+3+5..."
Command3.Caption = "Calcular serie 1*3*5..."
Command4.Caption = "Salir"
End Sub

Private Sub Command1_Click()
Call EntrarDato
End Sub

Private Sub Command2_Click()
If Dato <> 0 Then Call CalculaSuma(Dato)
End Sub



'Continúa…
```

Código versiones VB más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
'[PROGRAMA IMPARES02]

Option Explicit On
Public Class Form1
    Dim Dato As Integer

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.Text = "Impares02"
        LabelResultado.Font = New Font("Arial", 12,
FontStyle.Bold)
        LabelResultado.TextAlign =
ContentAlignment.MiddleCenter
        Button1.Text = "Introducir valor"
        Button2.Text = "Calcular serie 1+3+5..."
        Button3.Text = "Calcular serie 1*3*5..."
        Button4.Text = "Salir"
    End Sub

    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Call EntrarDato()
    End Sub

    Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
        If Dato <> 0 Then Call CalculaSuma(Dato)
    End Sub

    REM Continúa...
```

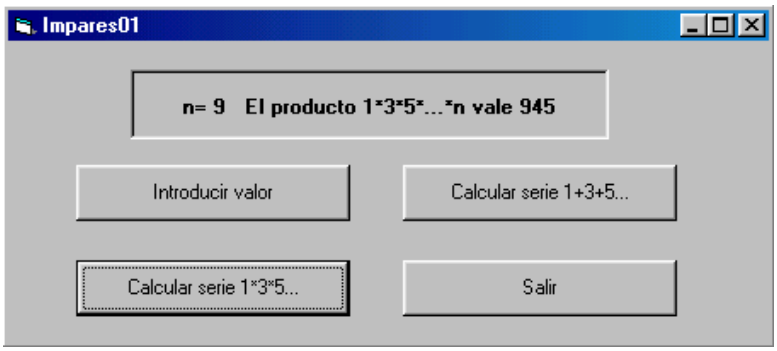
© 2006-2029 aprenderaprogramar.com

```
Private Sub Command3_Click()
If Dato <> 0 Then Call CalculaMult(Dato)
End Sub

Private Sub Command4_Click()
Unload Form1
End
End Sub

Private Sub EntrarDato()
Dim Par As Boolean
Dato = 0
Do While Dato <= 0 Or Dato > 19 Or Par = True
   Par = False
   Dato = Val(InputBox("Introduzca un número
impar entre 1 y 19", "Impar"))
   If Dato Mod 2 = 0 Then
      Par = True
      MsgBox("El número debe ser impar")
   End If
Loop
LabelResultado = "Dato aceptado"
End Sub

Private Sub CalculaSuma(ByVal Número As
Integer)
Dim Impar%, Suma As Integer
Impar = -1: Suma = 0
Do
   Impar = Impar + 2
   Suma = Suma + Impar
Loop While Impar < Número
LabelResultado = vbCrLf & "n= " & Dato & "    La
suma 1+3+5+...+n vale " & Suma
End Sub

Private Sub CalculaMult(ByVal Cifra As Integer)
Dim Impar As Integer
Dim Acumulado As Single
Impar = -1: Acumulado = 1
Do
   Impar = Impar + 2
   Acumulado = Acumulado * Impar
Loop While Impar < Cifra
LabelResultado = vbCrLf & "n= " & Dato & "    El
producto 1*3*5*...*n vale " & _ Acumulado
End Sub
```

```
Private Sub Button3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button3.Click
        If Dato <> 0 Then Call CalculaMult(Dato)
End Sub

Private Sub Button4_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button4.Click
        Me.Close()
        End
End Sub

Private Sub EntrarDato()
   Dim Par As Boolean
   Dato = 0
   Do While Dato <= 0 Or Dato > 19 Or Par = True
            Par = False
            Dato = Val(InputBox("Introduzca un
número impar entre 1 y 19", "Impar"))
            If Dato Mod 2 = 0 Then
            Par = True
            MsgBox("El número debe ser impar")
            End If
   Loop
   LabelResultado.Text = "Dato aceptado"
End Sub

Private Sub CalculaSuma(ByVal Número As Integer)
        Dim Impar, Suma As Integer
        Impar = -1 : Suma = 0
        Do
            Impar = Impar + 2
            Suma = Suma + Impar
        Loop While Impar < Número
        LabelResultado.Text = vbCrLf & "n= " &
Dato & "    La suma 1+3+5+...+n vale " & Suma
    End Sub

Private Sub CalculaMult(ByVal Cifra As Integer)
        Dim Impar As Integer
        Dim Acumulado As Single
        Impar = -1 : Acumulado = 1
        Do
            Impar = Impar + 2
            Acumulado = Acumulado * Impar
        Loop While Impar < Cifra
LabelResultado.Text = vbCrLf & "n= " & Dato   &
"  El producto 1*3*5*...*n vale " & Acumulado
End Sub

End Class
```

© 2006-2029 aprenderaprogramar.com

Aspecto gráfico de la ejecución del programa:



Próxima entrega: CU00346A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

EJERCICIO RESUELTO

Generar el código en Visual Basic correspondiente al siguiente pseudocódigo, relativo a un algoritmo para el manejo de una lista de datos sobre la que se desea tener opción a añadir un dato, borrar un dato, conocer el total de datos, visualizar la lista de datos o finalizar.

Pseudocódigo:

```
PROGRAMA DATOS01 [Curso Visual Basic aprenderaprogramar.com]

    Variables
        Enteras: E
        Reales: Dato()

    1. Inicio
        2. Mientras E <> 5 Hacer
            2.1 Si E = 1 Entonces
                    Llamar Añadir
                FinSi
            2.2 Si E = 2 Entonces
                    Llamar Borrar
                FinSi
            2.3 Si E = 3 Entonces
                    Mostrar "El número de datos actualmente en la lista es de",
                            Limitesuperior(Dato)
                FinSi
            2.4 Si E = 4 Entonces
                    Llamar Listar
                FinSi
            2.5 Mostrar "Elija una opción.  1. Añadir dato.
                                            2. Borrar dato.
                                            3. Conocer el total de datos.
                                            4. Mostrar Lista.
                                            5. Salir."
            2.6 Pedir E
        Repetir
    3. Fin

    [Continúa…]
```

© 2006-2029 aprenderaprogramar.com

```
Módulo Añadir
    1. Redimensionar Dato(Limitesuperior(Dato) + 1)
    2. Mostrar "Introduzca valor a añadir"
    3. Pedir Dato(Limitesuperior(Dato))
FinMódulo

Módulo Borrar
    Variables
        Enteras: i, Numborrar
    1. Si Limitesuperior(Dato) <> 0 Entonces
        1.1 Numborrar = 0
        1.2 Mientras Numborrar < 1 ó Numborrar > Limitesuperior(Dato) Hacer
            Mostrar "Introduzca el nº de dato a borrar"
            Pedir Numborrar
          Repetir
        1.3 Si Numborrar < Limitesuperior(Dato) Entonces
            Desde i = Numborrar hasta Limitesuperior(Dato) – 1
                Dato(i) = Dato(i + 1)
            Siguiente
          FinSi
        1.4 Redimensionar Dato(Limitesuperior(Dato) – 1)
        1.5 Mostrar "Dato borrado"
      FinSi
    2. Si Limitesuperior(Dato) = 0 Entonces
        Mostrar "La lista está vacía"
      FinSi
FinMódulo

Módulo Listar
    Variables
        Enteras: i
    1. Desde i = 1 hasta Limitesuperior(Dato) Hacer
        Mostrar Dato(i)
      Siguiente
FinMódulo
```

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN

Código versiones menos recientes VB:

```
'Curso Visual Basic aprenderaprogramar.com
'[PROGRAMA DATOS01]
Option Explicit
Dim Dato() As Single

Private Sub Form_Load()
Form1.Caption = "Datos01"
LabelResultado.Alignment = 2
LabelResultado.FontBold = True
LabelLista.Alignment = 2
LabelLista.FontBold = True
ReDim Dato(0)
Command1.Caption = "Añadir dato"
Command2.Caption = "Borrar dato"
Command3.Caption = "Conocer total de datos"
Command4.Caption = "Mostrar lista"
Command5.Caption = "Salir"
End Sub

Private Sub Command1_Click()
Call Añadir
End Sub
Private Sub Command2_Click()
Call Borrar
End Sub
Private Sub Command3_Click()
LabelResultado = "El número de datos
actualmente en la lista es de " & UBound(Dato)
End Sub
Private Sub Command4_Click()
Call Listar
End Sub
Private Sub Command5_Click()
Unload Form1
End
End Sub

Private Sub Añadir()
LabelResultado = " "
LabelLista = " "
ReDim Preserve Dato(UBound(Dato) + 1)
Dato(UBound(Dato)) =
Val(InputBox("Introduzca valor a añadir",
"Dato"))
End Sub

'Continúa…
```

Código versiones más recientes VB:

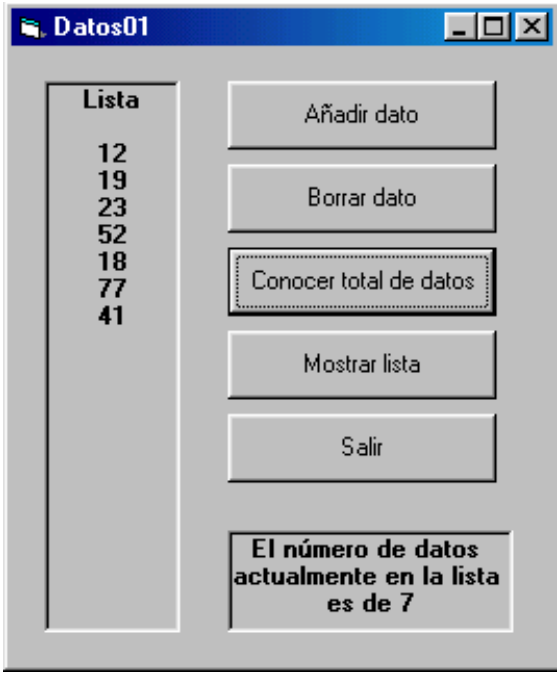
```
REM Curso Visual Basic aprenderaprogramar.com
'[PROGRAMA DATOS01]
Option Explicit On

Public Class Form1
    Dim Dato() As Single

Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.Text = "Datos01"
LabelResultado.Font = New Font("Arial", 12,
FontStyle.Bold)
LabelResultado.TextAlign = ContentAlignment.MiddleCenter
LabelLista.Font = New Font("Arial", 12, FontStyle.Bold)
LabelLista.TextAlign = ContentAlignment.MiddleCenter
        ReDim Dato(0)
        Button1.Text = "Añadir dato"
        Button2.Text = "Borrar dato"
        Button3.Text = "Conocer total de datos"
        Button4.Text = "Mostrar lista"
        Button5.Text = "Salir"
    End Sub

Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Call Añadir()
End Sub

Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        Call Borrar()
End Sub

Private Sub Button3_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button3.Click
        LabelResultado.Text = "El número de datos
actualmente en la lista es de " & UBound(Dato)
End Sub

Private Sub Button4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button4.Click
        Call Listar()
End Sub

Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button5.Click
        Me.Close()
        End
End Sub

Private Sub Añadir()
     LabelResultado.Text = " "
     LabelLista.Text = " "
     ReDim Preserve Dato(UBound(Dato) + 1)
     Dato(UBound(Dato)) = Val(InputBox("Introduzca
valor a añadir", "Dato"))
End Sub

'Continúa…
```

© 2006-2029 aprenderaprogramar.com

```
Private Sub Borrar()
Dim i%, Numborrar As Integer
LabelLista = " "
If UBound(Dato) <> 0 Then
   Numborrar = 0
   Do While Numborrar < 1 Or Numborrar >
UBound(Dato)
      Numborrar = InputBox("Introduzca el
número de dato a borrar", "Borrar")
   Loop
   If Numborrar < UBound(Dato) Then
      For i = Numborrar To UBound(Dato) - 1
         Dato(i) = Dato(i + 1)
      Next i
   End If
   ReDim Preserve Dato(UBound(Dato) - 1)
   LabelResultado = "Dato borrado"
End If
If UBound(Dato) = 0 Then
   LabelResultado = "La lista está vacía"
End If
End Sub

Private Sub Listar()
Dim i As Integer
LabelLista = "Lista" & vbCrLf
For i = 1 To UBound(Dato)
   LabelLista = LabelLista & vbCrLf & Dato(i)
Next i
End Sub
```

```
Private Sub Borrar()
        Dim i, Numborrar As Integer
        LabelLista.Text = " "
If UBound(Dato) <> 0 Then
           Numborrar = 0
           Do While Numborrar < 1 Or Numborrar >
UBound(Dato)
              Numborrar = InputBox("Introduzca el
número de dato a borrar", "Borrar")
           Loop
           If Numborrar < UBound(Dato) Then
             For i = Numborrar To UBound(Dato) - 1
                   Dato(i) = Dato(i + 1)
             Next i
           End If

           ReDim Preserve Dato(UBound(Dato) - 1)
           LabelResultado.Text = "Dato borrado"
        End If

        If UBound(Dato) = 0 Then
           LabelResultado.Text = "La lista está
vacía"
        End If
    End Sub

    Private Sub Listar()
        Dim i As Integer
        LabelLista.Text = "Lista" & vbCrLf
        For i = 1 To UBound(Dato)
           LabelLista.Text = LabelLista.Text &
vbCrLf & Dato(i)
        Next i
    End Sub
End Class
```

Aspecto gráfico del programa:



© 2006-2029 aprenderaprogramar.com

Comentarios: Visual Basic no admite pedir el límite superior de una matriz cuyo número de dimensiones no ha sido definido. Por ello introducimos la línea ReDim Dato(0) cuando se carga el programa. De este modo, con el primer dato a añadir ya podemos dar la orden de que se amplíe el límite superior del vector Dato en una unidad, asignándose el primer valor introducido a Dato(1). ¿Qué ocurre con Dato(0)? Simplemente se trata de un elemento auxiliar de un array que facilita la programación, sin otra funcionalidad. Se trata de una decisión que adoptamos como programadores, aunque podríamos haber usado el elemento de índice cero si hubiéramos querido.

Próxima entrega: CU00347A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

VERIFICACIÓN DE ALGORITMOS UTILIZANDO VISUAL BASIC

Cuando hablamos de "Verificación de algoritmos" en el curso “Bases de la programación nivel I” de aprenderaprogramar.com, distinguimos cinco técnicas que eran:

- Verificación mental.
- Verificación por seguimiento escrito.
- Verificación por seguimiento con tablas de variables.
- Verificación por seguimiento con desarrollo en un lenguaje.
- Verificación por seguimiento con un lenguaje y paso a paso.

Vamos a abordar cómo usar estas técnicas con Visual Basic. Dado que la verificación mental o por seguimiento escrito son propias del programador y su mente, o a lo sumo, un papel de ayuda, comenzaremos por ver cómo construir tablas de variables con Visual Basic.

CONSTRUIR TABLAS DE VARIABLES A TRAVÉS DE VISUAL BASIC

Consideremos el siguiente código, correspondiente a una estructura de bucle anidado dentro de otro:

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim i%, j As Integer
Dim A As Integer
Private Sub Form_Load()
Label1 = " "
For i = 1 To 5
   For j = 1 To 5
      A = i * j
      Label1 = Label1 & A & ", "
   Next j
Label1 = Label1 & " // "
Next i
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

        Dim i, j As Integer
        Dim A As Integer

        Label1.Text = ""

        For i = 1 To 5
            For j = 1 To 5
                A = i * j
                Label1.Text = Label1.Text & A & ", "
            Next j
            Label1.Text = Label1.Text & " // "
        Next i
    End Sub
End Class
```

Supongamos que queremos construir una tabla de variables en Visual Basic con 3 columnas:

Paso bucle 01 (i)	Paso bucle 02 (j)	Valor de A
.	.	.
.	.	.
.	.	.
.	.	.

© 2006-2029 aprenderaprogramar.com

Para ello pdemos escribir el siguiente código:

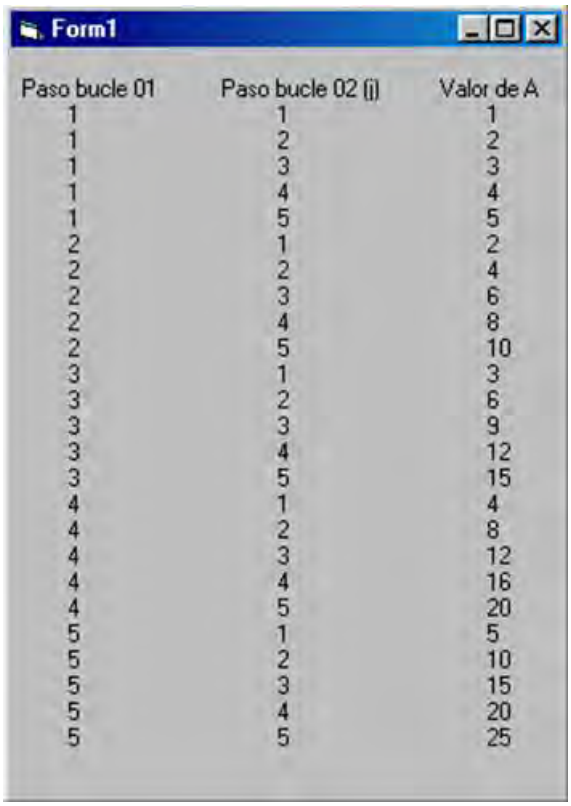
Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim i%, j As Integer
Dim A As Integer

Private Sub Form_Load()
Label1 = "": Label2 = "": Label3 = ""
Label1 = "Paso bucle 01" & vbCrLf
Label2 = "Paso bucle 02 (j)" & vbCrLf
Label3 = "Valor de A" & vbCrLf
For i = 1 To 5
   For j = 1 To 5
   A = i * j
   Label1 = Label1 & "    " & i & vbCrLf
   Label2 = Label2 & "    " & j & vbCrLf
   Label3 = Label3 & "    " & A & vbCrLf
   Next j
Next i
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

Dim i, j As Integer
Dim A As Integer

Label1.Text = "" : Label2.Text = "" : Label3.Text = ""
Label1.Text = "Paso bucle 01" & vbCrLf
Label2.Text = "Paso bucle 02 (j)" & vbCrLf
Label3.Text = "Valor de A" & vbCrLf
For i = 1 To 5
    For j = 1 To 5
     A = i * j
     Label1.Text = Label1.Text & "     " & i & vbCrLf
     Label2.Text = Label2.Text & "     " & j & vbCrLf
     Label3.Text = Label3.Text & "     " & A & vbCrLf
    Next j
Next i
End Sub
End Class
```

Con un resultado gráfico tal como este:



© 2006-2029 aprenderaprogramar.com

Hemos utilizado para mostrar un resultado un Label, pero también podríamos haberlo hecho sobre un TextBox. ¿Por qué? No hay motivo por el cual usar un Label sea obligatorio. Cada cual lo hará como lo estime conveniente. Nosotros usamos normalmente Labels para mostrar resultados, pero en algunos casos, por ejemplo si nos interesa poder copiar y pegar un texto, usamos TextBox.

Supongamos un programa que contiene el doble bucle que hemos visto. Hemos querido verificarlo y lo hemos hecho con esta visualización. Una vez estamos satisfechos del resultado del bucle, borramos la verificación que hemos utilizado para visualizar y seguimos trabajando tranquilos, sabemos que el bucle no va a producir efectos indeseados.

En ocasiones nos interesa obtener un resultado en forma de tabla de variables que podamos exportar por ejemplo a un procesador de textos. Una forma de hacerlo sería programándolo, pero también podemos hacer que los resultados se presenten sobre un TextBox como hemos indicado antes y a partir de ahí lo único que tenemos que hacer es copiar (Ctrl + C) y pegar (Ctrl + V). Vamos a ver este caso en código. Primeramente crearemos un TextBox y en la ventana de propiedades establecemos Multiline = True y Scrollbars = 3 (Both).

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim i%, j As Integer
Dim A As Integer
Dim Espacios As String

Private Sub Form_Load()
Espacios = "                    "
Text1 = "Paso bucle 01 " & " Paso bucle
02 (j) " & " Valor de A" & vbCrLf
For i = 1 To 5
  For j = 1 To 5
    A = i * j
    Text1 = Text1 & "        " & i &
Espacios & j & Espacios & A & vbCrLf
  Next j
Next i
End Sub
```

Código versiones más recientes VB

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

        Dim i, j As Integer
        Dim A As Integer
        Dim Espacios As String

        Espacios = "                          "
        TextBox1.Text = "Paso bucle 01 " & " Paso bucle
02 (j) " & " Valor de A" & vbCrLf
        For i = 1 To 5
         For j = 1 To 5
         A = i * j
         TextBox1.Text = TextBox1.Text & "           " &
i & Espacios & j & Espacios & A & vbCrLf
            Next j
        Next i
    End Sub
End Class
```

Si Multiline no está establecido en True nos saldrá el resultado en una sola línea. El resultado es la ya conocida tabla, pero con este código en formato "capturable" en el portapapeles, de modo que podemos por ejemplo pasarlo a una hoja de cálculo.

Próxima entrega: CU00348A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

PROGRAMACIÓN RÁPIDA. VERIFICACIÓN DE ALGORITMOS EN VISUAL BASIC.

La programación rápida busca verificar el funcionamiento de los algoritmos antes de entrar en aspectos más formales, como pueden ser la estética, comentarios, etc. La programación rápida puede ser entendida de distintas maneras según los criterios de cada programador. Algunas prácticas de programación rápida pueden ser las siguientes:

- Se usa el formato de salida por pantalla más sencillo posible. En nuestro caso usaremos Labels, TextBox ó msgBox.

- No se piden datos al usuario. La entrada de datos es controlada directamente por el programador, bien usando la instrucción Rnd para generar datos aleatorios o disponiendo una entrada de valores a variables directa, controlada por el programador.

- La estructura del programa no es la definitiva, con los diferentes módulos o procedimientos que pueda llevar, sino provisional.

Vamos a ver un ejemplo de lo que sería programación rápida aplicada al código que se muestra a continuación (Programa SUC02), relativo al cálculo de un sumatorio del tipo:

$$\sqrt{a},\ \sqrt{a-1},\ \sqrt{a-2},\ \ldots,\ \sqrt{0}$$

© 2006-2029 aprenderaprogramar.com

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
'Programa SUC02

Option Explicit
Dim Dato As Single
Dim Raíz01!, Raíz02 As Single
Dim Suce As Single

Private Sub Form_Load()
Form1.Caption = "Programa SUC02"
CommandCálculo.Caption = "Cálculo"
CommandSalir.Caption = "Salir"
End Sub

Private Sub CommandSalir_Click()
Unload Form1
End
End Sub

Private Sub CommandCálculo_Click()
Call EntraDatos
Call Proceso(Dato)
Call Resultados
End Sub

Private Sub EntraDatos()
Do
   Dato = CInt(InputBox("Por favor, introduzca
nº entero entre 0 y 100", "Dato"))
Loop While Dato < 0 Or Dato > 100
End Sub

Private Sub Proceso(ByVal Num As
Integer)
Raíz01 = Sqr(Num)
Raíz02 = -Raíz01
Suce = 0
Do While Num >= 0
   Suce = Suce + Sqr(Num)
   Num = Num - 1
Loop
End Sub

Private Sub Resultados()
Label1.Alignment = 2
Label1.FontBold = True
Label1 = "Dato base = " & Dato & vbCrLf &
vbCrLf & " Raíz01 = " & Raíz01 & " Raíz02 =" &
Raíz02 & vbCrLf & vbCrLf & "Valor sucesión = "
& Suce
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
'Programa SUC02

Option Explicit On
Public Class Form1
    Dim Dato As Single
    Dim Raíz01 As Single, Raíz02 As Single
    Dim Suce As Single

Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.Text = "Programa SUC02"
        ButtonCálculo.Text = "Cálculo"
        ButtonSalir.Text = "Salir"
        Label1.Text = ""
    End Sub

    Private Sub ButtonSalir_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ButtonSalir.Click
        Me.Close()
        End
    End Sub

    Private Sub ButtonCálculo_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ButtonCálculo.Click
        Call EntraDatos()
        Call Proceso(Dato)
        Call Resultados()
    End Sub

    Private Sub EntraDatos()
        Do
            Dato = CInt(InputBox("Por favor,
introduzca nº entero entre 0 y 100", "Dato"))
        Loop While Dato < 0 Or Dato > 100
    End Sub

    Private Sub Proceso(ByVal Num As Integer)
        Raíz01 = Math.Sqrt(Num)
        Raíz02 = -Raíz01
        Suce = 0
        Do While Num >= 0
            Suce = Suce + Math.Sqrt(Num)
            Num = Num - 1
        Loop
    End Sub

    Private Sub Resultados()
 Label1.TextAlign = ContentAlignment.MiddleCenter
 Label1.Font = New Font("Arial", 10, FontStyle.Bold)
 Label1.Text = "Dato base = " & Dato & vbCrLf &
vbCrLf & " Raíz01 = " & Raíz01 & " Raíz02 =" &
Raíz02 & vbCrLf & vbCrLf & "Valor sucesión = " &
Suce
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN CÓDIGO DE PROGRAMACIÓN RÁPIDA

Código versiones menos recientes VB:

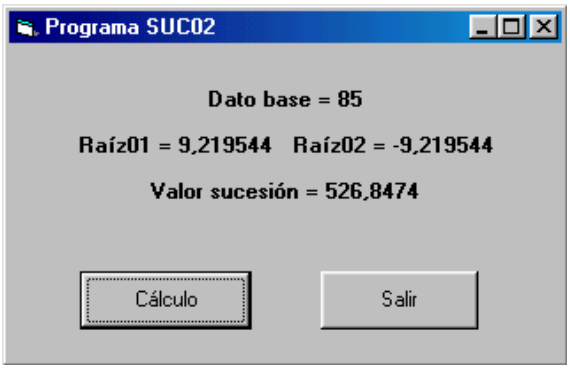
```
'Curso VB aprenderaprogramar.com
'Preliminar SUC02

Option Explicit
Dim Datobase!, Dato!, Raíz01!, Raíz02!,
Suce As Single

Private Sub Form_Load()
Label1 = “”
Datobase = 85   '[Línea de introducción
manual de datos]
Rem Randomize: Datobase = Cint(Rnd *
100 + 1)   '[Línea de generación _
aleatoria de datos, actualmente
desactivada]
Dato = Datobase
Raíz01 = Sqr(Dato)
Raíz02 = -Raíz01
Suce = 0
Do While Dato >= 0
   Suce = Suce + Sqr(Dato)
   Dato = Dato - 1
Loop
Label1 = Label1 & "Dato base = " &
Datobase & vbCrLf & "Raíz01 = " & Raíz01
& "    Raíz02 = " & Raíz02 & vbCrLf & "
Valor sucesión = " & Suce
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
'Preliminar SUC02

Option Explicit On
Public Class Form1

    Dim Datobase, Dato, Raíz01, Raíz02, Suce As Single

Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

Label1.Text = ""
Datobase = 85   '[Línea introducción manual de datos]
REM  generación aleatoria de datos desactivada]
'Randomize() : Datobase = CInt(Rnd() * 100 + 1)

        Dato = Datobase
        Raíz01 = Math.Sqrt(Dato)
        Raíz02 = -Raíz01
        Suce = 0
        Do While Dato >= 0
            Suce = Suce + Math.Sqrt(Dato)
            Dato = Dato - 1
        Loop
        Label1.Text = Label1.Text & "Dato base = " &
Datobase & vbCrLf & "Raíz01 = " & Raíz01 & "    Raíz02
= " & Raíz02 & vbCrLf & " Valor sucesión = " & Suce
    End Sub
End Class
```

El resultado de este código es coincidente con el que obtuvimos en el ejercicio de referencia donde creamos el programa, es decir:

Datobase = 85
Raíz 01 = 9,219544 Raíz 02 = -9,219544
Valorsucesión = 526,8474



La diferencia respecto al código que se propuso en el citado ejercicio está en que:

- Se han eliminado aspectos relacionados con la estética del programa como que se muestre un título de formulario "Programa SUC02". Ya en los programas que venimos desarrollando la atención a

© 2006-2029 aprenderaprogramar.com

cuestiones estéticas es más bien escasa, pero para la verificación de un algoritmo nos olvidamos completamente de ellas.

- La estructura del programa, que constaba de 2 opciones, Cálculo y Salir, con 3 procedimientos asociados a la opción cálculo (EntraDatos, Proceso y Resultados) se ha eliminado, existiendo un único código asociado a la carga del formulario por ser lo más rápido.
- La petición de datos al usuario con un InputBox de tipo (Dato = CInt(InputBox("Por favor, introduzca nº entero entre 0 y 100", "Dato")) se ha eliminado. A cambio, existen dos líneas con las que el programador trabaja. Una en la que introduce datos manualmente y otra en las que genera datos aleatoriamente. De este modo puede hacer tanto comprobaciones específicas como comprobaciones aleatorias.
- Las variables que intervienen no coinciden exactamente con las del programa definitivo.

El programador introducirá los cambios que le resulten de interés para la verificación del programa. Por ejemplo, será habitual introducir un botón para que cada vez que sea pulsado se repita la muestra de resultados (generando datos aleatorios). Para ello nos basta desactivar la introducción de datos manual, activar la aleatoria e introducir un botón que se encargue de recargar el formulario. El código sería el siguiente:

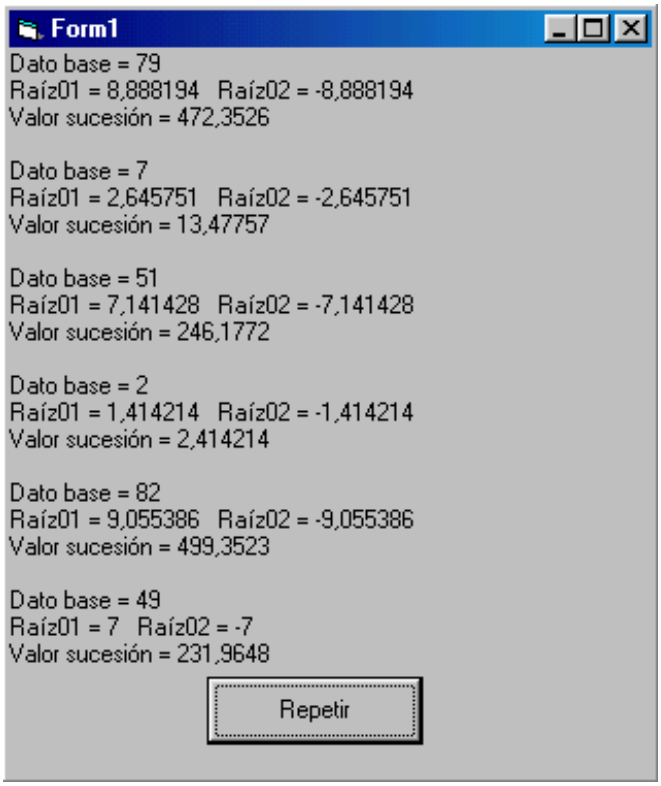
Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Private Sub CommandRepetir_Click()
Form_Load
End Sub
```

Código versiones más recientes VB

```
REM Curso Visual Basic aprenderaprogramar.com
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Me.Form1_Load(Me, New System.EventArgs)
End Sub
```

De este modo podemos llegar a resultados como el que se muestra a continuación:



© 2006-2029 aprenderaprogramar.com

De una pantalla de este tipo un programador avezado puede obtener numerosa información respecto al funcionamiento del algoritmo. Fijémonos por ejemplo en los resultados para dato base = 49. Las raíces que nos da son 7 y -7. De ahí podemos intuir que el cálculo de raíces se está realizando correctamente. La raíz esperada para 81 sería 9. Tenemos un dato base 82 con raíz 9,05, luego con distintos datos parece que el cálculo de raíces es correcto. En cuanto a los sumatorios, podríamos compararlos con datos de que dispongamos previamente para valorar su corrección, o incluso valorarlos comparativamente. Podemos esperar que el sumatorio para un dato base 79 sea similar (ligeramente inferior) al de un dato base 82. Comprobamos que es así.

Si para un dato base 49 se nos mostraran raíces 5,05 y -5,05, o el sumatorio correspondiente a 79 fuera superior a 82, estaríamos ante una situación de no superación de la verificación. Si se diera esta circunstancia, tendríamos que apuntar los datos que nos están dando lugar a resultados erróneos, revisar el algoritmo por si existen errores de sintaxis, trascripción, etc. e incluso modificar su diseño si fuera necesario. Por último, cuando creamos que está funcionando razonablemente, introduciremos de forma manual los datos que apuntamos en su momento como generadores de errores y comprobaremos que para esos datos no hay problema ninguno.

Las propuestas de verificación que hemos visto se corresponden con la verificación funcional del algoritmo, ya que mediante un número limitado de pruebas inferimos que su funcionamiento es correcto. A su vez, para la entrada manual se supone que escogemos una serie de casos representativos, lo que se correspondería con lo que hemos denominado verificación razonada. Por el contrario, la ejecución del programa con datos de entrada generados con Randomize y Rnd se correspondería con una verificación aleatoria.

La verificación total de este algoritmo consistiría en un programa que nos mostrara los resultados para todos los valores posibles (enteros comprendidos entre 0 y 100), de forma que pudiéramos comprobar la corrección de cada uno de ellos. Esto sería posible, pero en general supone un esfuerzo que se considera no necesario ya que las verificaciones funcionales bien realizadas resultan rápidas y efectivas. Además, en muchos casos una verificación completa es inviable, ya que en muchos problemas los posibles datos de entrada son infinitos (por ejemplo, los números enteros).

Próxima entrega: CU00349A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

PASO A PASO Y PUNTOS DE INTERRUPCIÓN (BREAKPOINTS)

Visual Basic dispone de distintas opciones relacionadas con la ejecución paso a paso de un programa, de las cuales vamos a ver una de ellas: **los puntos de interrupción**. Se entiende por punto de interrupción una línea del programa donde el programador decide detener la ejecución. El objetivo será comprobar qué valor tienen las variables, qué resultados se han obtenido por pantalla hasta el momento, etc. y de esta forma comprobar si el programa hasta ese momento está haciendo lo que se espera.

Una vez **posicionado el cursor sobre la línea** en que queremos detener el programa, hay cuatro maneras de establecer puntos de interrupción:

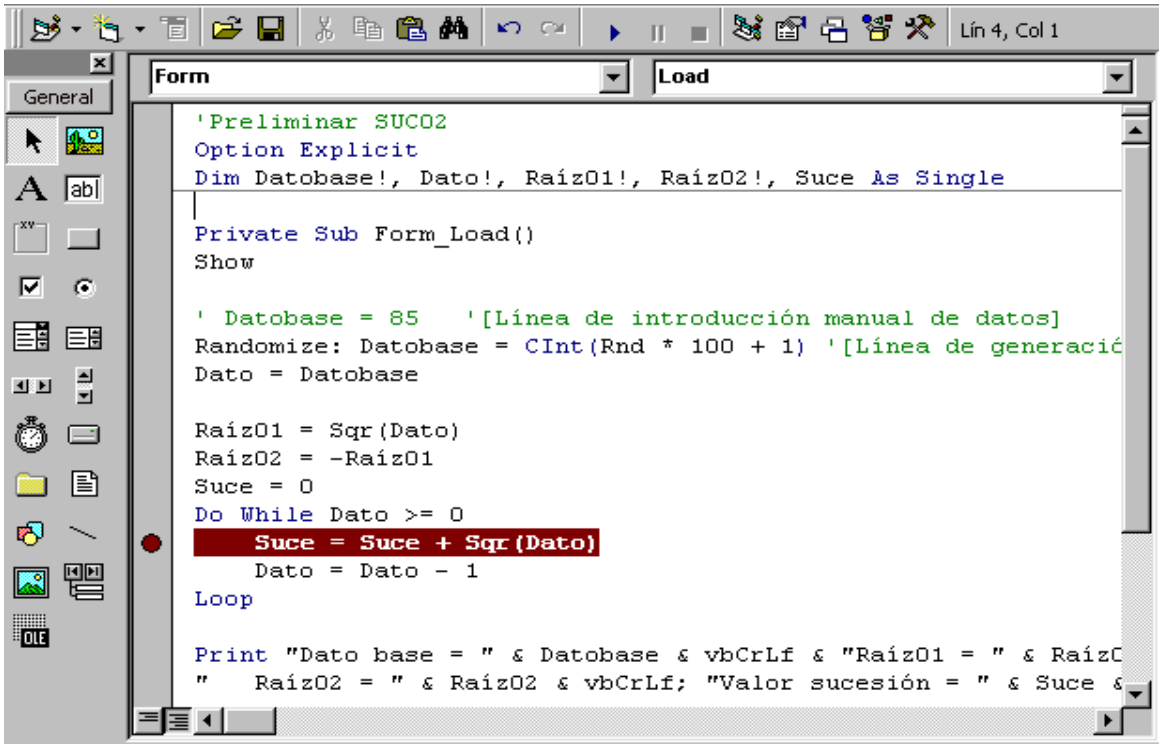
a) En el menú Depuración (Depurar), elige la opción alternar puntos de interrupción.

b) Pulsa F9.

c) Pulsa el botón derecho del ratón y elige Punto de interrupción – Insertar punto de interrupción ó Alternar - Puntos de interrupción (según versión de Visual Basic que estés utilizando).

d) Sobre la banda gris que está en el lateral izquierdo de la ventana de código, pincha con el ratón a la altura de la línea en que quieres establecer un punto de interrupción.

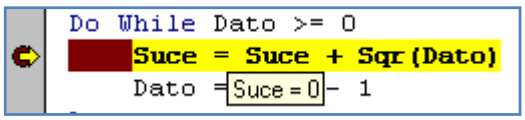
En cualquiera de los casos, el resultado será que la línea elegida aparecerá subrayada en rojo y con un punto rojo en su lado izquierdo. El aspecto del editor será similar al mostrado a continuación (aunque depende de la versión de Visual Basic que estemos utilizando).



© 2006-2029 aprenderaprogramar.com

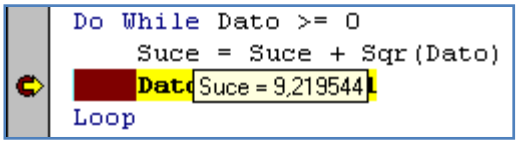
Vamos a suponer que estamos trabajando con el programa 'Preliminar SUC02' cuyo código hemos visto en la entrega anterior de este curso de Visual Basic de aprenderaprogramar.com, y que establecemos un punto de interrupción o breakpoint en la línea Suce = Suce + Math.Sqrt(dato) ó Suce = Suce + Sqr(Dato) según versión, dentro del bucle Do While.

Si ejecutas el programa (p.ej. pulsando F5), comprobarás que el programa se detiene, indicando con una flecha amarilla que se encuentra detenido al llegar a la línea que hemos indicado como punto de interrupción. Comprueba ahora cómo si posicionas el puntero del ratón sobre una variable, por ejemplo Suce, Visual Basic te muestra el valor que tiene la variable Suce justo en ese momento.



Para continuar ejecutando el programa se ha de pulsar el icono de ejecución (o pulsar F5 nuevamente), y el programa continuará hasta encontrarse nuevamente con un punto de interrupción. Ten en cuenta que un punto de interrupción dentro de un bucle detendrá el programa tantas veces como se ejecute el bucle.

Igual que para variables situadas sobre la línea de interrupción, puedes consultar el valor de variables en otras líneas. Ten en cuenta que el valor mostrado es siempre el valor de las variables antes de ejecutar la línea de interrupción. Por eso en la primera interrupción nos indica que Suce vale 0. Si quisiéramos saber el valor de las variables justo después de ejecutarse esa línea, es decir, qué valor ha adquirido Suce, podemos hacerlo de dos maneras. La primera, sumando los valores actuales a la derecha de la igualdad. La segunda, estableciendo la interrupción en la línea inmediatamente posterior a aquella línea que queremos evaluar. En nuestro caso podríamos establecer la interrupción en la línea Dato = Dato – 1 de modo que en esta línea podemos comprobar el valor que ha tomado Suce después de ejecutada la línea anterior:



Para un dato base igual a 85, el primer valor de Suce tras ejecutarse la línea es 9,219544. Para realizar este tipo de interpretación hemos establecido la interrupción en la línea inmediata posterior a aquella que nos resulta de interés. Comprueba cómo va variando el valor de Suce con cada paso por el bucle y razona si los valores que va tomando son correctos o no.

Se pueden establecer tantos puntos de interrupción como se estimen oportunos. Normalmente irán colocados en lugares que el programador estima estratégicos para sus intenciones como pueden ser: antes de entrar a un bucle, en el interior de un bucle, tras salir de un bucle, etc.

© 2006-2029 aprenderaprogramar.com

Un punto de interrupción se elimina de la misma manera que se puso: si estando sobre una línea marcada pulsamos F9 o elegimos alternar puntos de interrupción, será desactivada. También podemos suprimir todos los puntos de interrupción existentes a través de una opción más avanzada.

Existen otras formas de paso a paso con Visual Basic como son: Paso a Paso por instrucciones (F8), Paso a Paso por Procedimientos (Mayús + F8), o Ejecutar hasta el cursor (Ctrl + F8). Existen muchas más opciones de configuración y posibilidades para la depuración de programas con Visual Basic. Ten en cuenta que con aplicaciones profesionales, largas y extensas, pueden existir miles de líneas de código. Los programadores profesionales utilizan las herramientas de depuración disponibles para detectar y corregir errores en el código.

Próxima entrega: CU00350A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

DEPURAR ALGORITMOS

Vamos a reflejar en forma de código enfoques para la verificación de algoritmos. Partimos del siguiente bucle ejemplo:

```
For i = a To b
   For j = m To n
      R = i * j
   Next j
Next i
```

Los enfoques para depurar el algoritmo los podemos establecer a partir de puntos de interrupción. Sin embargo, en general es más rápido hacer una simple constatación del valor de variables que establecer una interrupción del programa, ya que por cada parada estaremos perdiendo un tiempo. Con este razonamiento, usaremos un Label para dejar constancia del valor de una variable en un punto dado del programa. Sólo cuando no seamos capaces de llegar a conclusiones sin tener que detener el programa, estableceremos puntos de interrupción.

- **Enfoque al resultado**

Nos centramos en comprobar un valor final, en este caso R, suponiendo que si este valor es correcto el resto de procesos también lo son. Por ejemplo:

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim i%, j%
Dim a%, b%
Dim m%, n%
Dim R%

Private Sub Form_Load()
Label1 = ""
a = 1: b = 3: m = 1: n = 3
For i = a To b
   For j = m To n
      R = i * j
      Label1 = Label1 & R & vbCrLf
   Next j
Next i
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Dim i%, j%
        Dim a%, b%
        Dim m%, n%
        Dim R%
        Label1.Text = ""
        a = 1 : b = 3 : m = 1 : n = 3
        For i = a To b
            For j = m To n
                R = i * j
                Label1.Text = Label1.Text & R & vbCrLf
            Next j
        Next i
    End Sub
End Class
```

En el programa anterior hemos verificado qué valor toma R en cada pasada del bucle. El objetivo del bucle va a ser transferir el valor de R a otros procesos, pero antes de hacer eso hemos querido comprobar qué resultado era el que obteníamos con este diseño de bucle.

© 2006-2029 aprenderaprogramar.com

- **Enfoque al proceso**

Analizamos qué ocurre en uno o varios pasos del programa que resultan de nuestro interés, estudiando los valores que van tomando las variables. Por ejemplo:

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim i%, j%
Dim a%, b%
Dim m%, n%

Private Sub Form_Load()
Label1 = ""
a = 1: b = 3: m = 1: n = 3
Label1 = Label1 & "a =" & a & " b =" & b & vbCrLf
For i = a To b
   Label1 = Label1 & "m =" & m & " n =" & n & " i =" & i & vbCrLf
   For j = m To n
      Label1 = Label1 & "j =" & j & vbCrLf
   Next j
Next i
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

  Dim i%, j%
  Dim a%, b%
  Dim m%, n%

Label1.Text = ""
a = 1 : b = 3 : m = 1 : n = 3
Label1.Text = Label1.Text & "a =" & a & " b =" & b &
vbCrLf
For i = a To b
    Label1.Text = Label1.Text & "m =" & m & " n =" & n
& " i =" & i & vbCrLf
    For j = m To n
        Label1.Text = Label1.Text & "j =" & j & vbCrLf
    Next j
Next i
End Sub
End Class
```

Este programa da lugar a que se nos muestre lo siguiente:

a = 1 b = 3
m = 1 n = 3 i = 1
j = 1
j = 2
j = 3
m = 1 n = 3 i = 2
j = 1
j = 2
j = 3
m = 1 n = 3 i = 3
j = 1
j = 2
j = 3

A partir de estos valores de variables el programador analiza si el programa está respondiendo a lo que él tenía previsto o si se está produciendo un resultado indeseado. En caso de detectar algún error, procede a la corrección y a una nueva prueba.

© 2006-2029 aprenderaprogramar.com

- **Enfoques zonales**

Analizamos qué ocurre en una zona de código donde estimamos ocurren procesos de interés o donde pensamos que puede existir un problema. Siguiendo con nuestro ejemplo, un enfoque zonal a la entrada del bucle sería:

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit

Dim i%, j%
Dim a%, b%, m%, n%, R%

Private Sub Form_Load()
Label1 = ""
a = 1: b = 3: m = 1: n = 3
Label1 = Label1 & "a = " & a & " b = " &
b & vbCrLf
Label1 = Label1 & "m = " & m & " n = " &
n & vbCrLf
Label1 = Label1 & "i = " & i & " j = " & j &
vbCrLf
Label1 = Label1 & "R = " & R & vbCrLf
For i = a To b
   For j = m To n
      R = i * j
      Label1 = Label1 & R & " ,"
   Next j
Next i
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

        Dim i%, j%
        Dim a%, b%, m%, n%, R%

        Label1.Text = ""
        a = 1 : b = 3 : m = 1 : n = 3
        Label1.Text = Label1.Text & "a = " & a & " b =
" & b & vbCrLf
        Label1.Text = Label1.Text & "m = " & m & " n =
" & n & vbCrLf
        Label1.Text = Label1.Text & "i = " & i & " j =
" & j & vbCrLf
        Label1.Text = Label1.Text & "R = " & R &
vbCrLf
        For i = a To b
            For j = m To n
                R = i * j
                Label1.Text = Label1.Text & R & " ,"
            Next j
        Next i
    End Sub
End Class
```

El resultado obtenido es:

a = 1 b = 3
m = 1 n = 3
i = 0 j = 0
R = 0
1, 2, 3, 2, 4, 6, 3, 6, 9

Se han verificado todas las variables a la entrada del bucle. Algunas no tiene demasiada lógica que se verifiquen, pero a veces los errores saltan donde menos son de esperar, y toda información es valiosa.

Próxima entrega: CU00351A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

EJERCICIO RESUELTO

Se ha previsto un programa donde se incluye un bucle que se pretende probar para comprobar que su funcionamiento es correcto. Para los valores emp = – 1, emp = 0, emp = 1, emp = 4, emp = 7 y emp = 10 construir las tablas de variables que describen el funcionamiento del siguiente bucle:

```
i = emp
Mientras i <= 7 Hacer
        i = i + 2
        Mostrar “2 * i vale”, 2 * i
Repetir
```

SOLUCIÓN

Versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim i%, emp%, iteración%

Private Sub Form_Load()
Label1 = ""
emp = Val(InputBox("Valor de emp="))
i = emp: iteración = 0
Label1 = Label1 & "Previo a bucle i =" & i &
vbCrLf
Do While i <= 7
iteración = iteración + 1
i = i + 2
Label1 = Label1 & " iteración =" & iteración &
" i =" & i & " Visualización:" & 2 * i & vbCrLf
Loop
End Sub

Private Sub BotonRepetir_Click()
Form_Load
End Sub
```

Versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
 Dim i%, emp%, iteración%
 Label1.Text = ""
 emp = Val(InputBox("Valor de emp="))
 i = emp : iteración = 0
 Label1.Text = Label1.Text & "Previo a bucle i =" &
i & vbCrLf
 Do While i <= 7
     iteración = iteración + 1
     i = i + 2
     Label1.Text = Label1.Text & " iteración =" &
iteración & " i =" & i & " Visualización:" & 2 * i
& vbCrLf
 Loop
End Sub

Private Sub BotonRepetir_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
BotonRepetir.Click
        Me.Form1_Load(Me, New System.EventArgs)
End Sub
End Class
```

El programa nos permite visualizar tantas tablas de variables como queramos pulsando un botón (button). En cada ocasión se nos pide el valor de emp a través de un InputBox.

Comprueba los resultados consultando las siguientes tablas.

© 2006-2029 aprenderaprogramar.com

Tablas:

emp = *−1*		
Estado	**i**	**Visualización**
Previo a bucle	*−1*	
Iteración 1	1	2
Iteración 2	3	6
Iteración 3	5	10
Iteración 4	7	14
Iteración 5	9	18

emp = *0*		
Estado	**i**	**Visualización**
Previo a bucle	*0*	
Iteración 1	2	4
Iteración 2	4	8
Iteración 3	6	12
Iteración 4	8	16

emp = *1*		
Estado	**i**	**Visualización**
Previo a bucle	1	
Iteración 1	3	6
Iteración 2	5	10
Iteración 3	7	14
Iteración 4	9	18

© 2006-2029 aprenderaprogramar.com

emp = *4*		
Estado	**i**	**Visualización**
Previo a bucle	*4*	
Iteración 1	6	12
Iteración 2	8	16

emp = *7*		
Estado	**i**	**Visualización**
Previo a bucle	*7*	
Iteración 1	9	18

emp = *10*		
Estado	**i**	**Visualización**
Previo a bucle	*10*	
—	—	— [No entra al bucle]

Próxima entrega: CU00352A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

ERRORES

Si has venido trabajando los diferentes programas y ejercicios que hemos planteado a lo largo del curso se te habrán presentado numerosos mensajes de error. Los mensajes de errores interceptados por Visual Basic son de dos tipos: "Errores de compilación" y "Errores de ejecución", coincidentes con la clasificación de errores que se hace en el curso “Bases de la programación nivel II” de aprenderaprogramar.com.

Ya hemos hablado algo de errores de sintaxis, y de la ayuda que nos proporciona Visual Basic a través de la corrección automática de errores. Los errores de sintaxis serán muy frecuentes y quizás incómodos, pues casi siempre nos encontraremos con uno o varios al tratar de ejecutar un programa. Sin embargo, son los menos "peligrosos", ya que de su fácil detección se deriva su corrección segura. Prueba a ejecutar este código donde existe un error de sintaxis (Lbel en vez de Label):

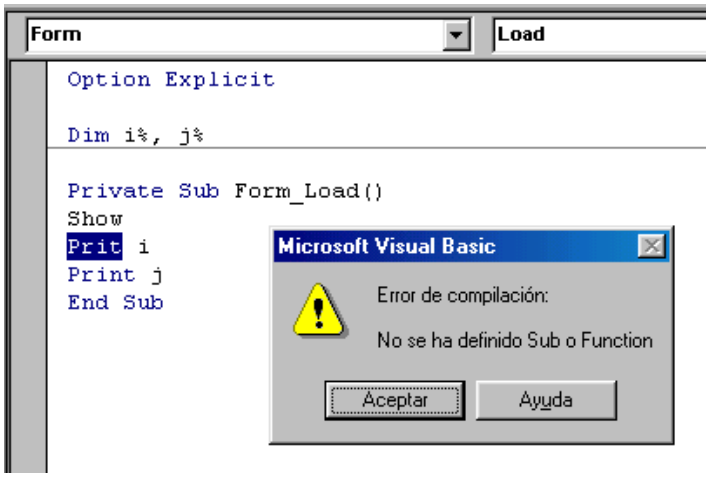
Código versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim i%, j%

Private Sub Form_Load()
Lbel1 = i
Label1 = Label1 & j
End Sub
```

Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Dim i%, j%
        Lbel1.text = i
        Label1.Text = Label1.Text & j
    End Sub
End Class
```

Al tratar de ejecutar el código te aparecerá un mensaje como “Error de compilación: variable no definida” ó “No se ha definido Sub o Function” ó “Lbl1 no está declarado” ó similar.



No se indica directamente que se trata de un error de sintaxis, ya que en algunas ocasiones a Visual Basic no le resulta fácil determinar si el error es de sintaxis o de otro tipo. En este caso por ejemplo, se interpreta que se está invocando una variable o procedimiento de nombre Lbl y te indica que dicho procedimiento o variable no está definida y por tanto no se puede invocar. Te aparece subrayado el término que está dando lugar al error, con lo cual rápidamente podrás comprobar que se trata de una escritura indebida y proceder a su corrección.

© 2006-2029 aprenderaprogramar.com

Los errores por procesos no válidos son en general detectados al tratar de ejecutar los programas, indicándonos Visual Basic qué tipo de error se ha producido, y en qué línea ha tenido lugar, aunque a veces no se detectan y dan lugar a "resultados extraños". Prueba a ejecutar el siguiente código:

Código versiones menos recientes de VB:

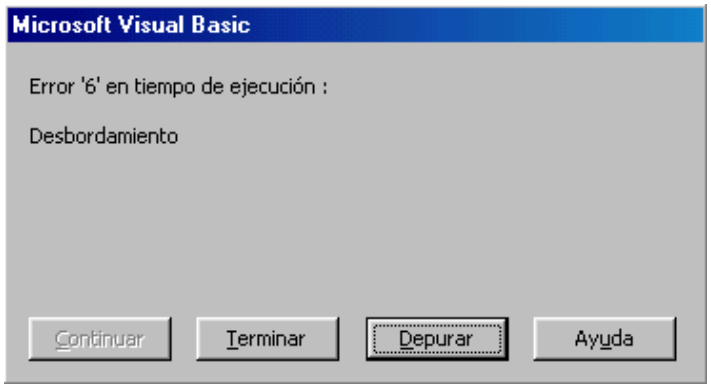
```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim i%, j%

Private Sub Form_Load()
Show
Label1 = i
Label1 = j
Label1 = i / j
End Sub
```

Código versiones más recientes de VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Dim i%, j%
        Lbel1.text = i
        Label1.Text = Label1.Text & j
    End Sub
End Class
```

Al llegar a la línea donde se trata de mostrar el valor de i / j el ordenador se encuentra con una indeterminación matemática ya que cero dividido entre cero no es un valor calculable. Esto puede dar lugar a dos tipos de situaciones:

a) La detención del programa y aparición de un mensaje de error.

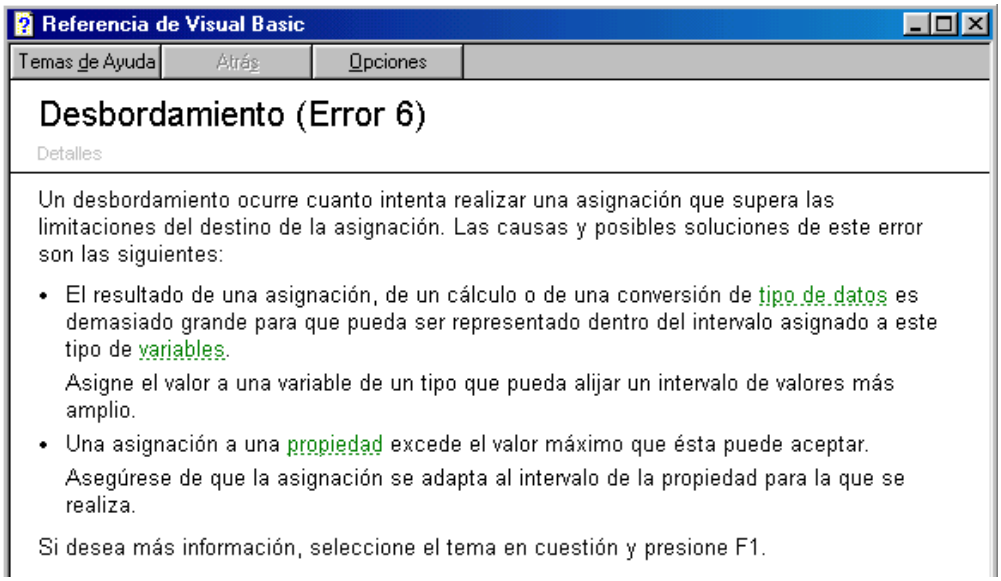


b) La obtención de un resultado extraño, por ejemplo que por pantalla aparezca NeuN (este valor extraño indica "resultado matemáticamente no determinable" y posiblemente termine generando un error y detención de un programa cuando surge en programas largos que utilizan este resultado para realizar más operaciones adicionales).

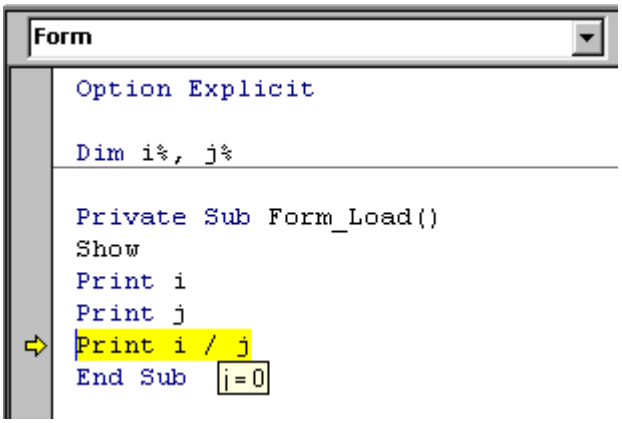
Cuando la ejecución de un programa se detiene debido a la aparición de un error Visual Basic nos suele ofrecer ayuda o información sobre el error que nos sirva para realizar las correcciones oportunas. Una vez detenido el programa podemos pulsar el botón "Terminar" para concluir la ejecución o también podremos pulsar en ayuda o sobre el error en una lista de errores para obtener ayuda e información adicional. La opción de ayuda nos conducirá a una descripción del tipo de error, recomendaciones para evitarlo, etc.

© 2006-2029 aprenderaprogramar.com

Podemos intervenir sobre el código para tratar de subsanar el error, con posibilidad de proseguir posteriormente la ejecución (pulsando F5). Visual Basic nos resalta y señala con una flecha la línea en que se ha producido la detención del programa, lo cual nos sirve de ayuda para corregir el error.



Cuando el programa está detenido podemos obtener información posicionando el cursor sobre un elemento del código (por ejemplo para conocer el valor que tiene una variable o el tipo de una variable).



No vamos a extendernos sobre las posibilidades que sobre errores ofrece Visual Basic, como programadores deberemos ir conociéndolas poco a poco a medida que tengamos más experiencia.

En relación a errores lógicos, las posibilidades son muy variadas. Algunos errores lógicos pueden ser detectados por Visual Basic, pero no conviene tener demasiadas expectativas al respecto. No te vamos a pedir que pruebes el siguiente código ya que puede bloquear el computador:

© 2006-2029 aprenderaprogramar.com

Código versiones menos recientes de VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit

Private Sub Form_Load()
Do

Loop

End Sub
```

Código versiones más recientes de VB

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Do

        Loop
    End Sub
End Class
```

Pero si por casualidad estás construyendo un programa y te has dejado olvidado o "sin solucionar" un bucle de este tipo, lo que puedes esperar es que el ordenador se quede "colgado" (bloqueado reptiendo un bucle continuamente). Con un poco de suerte podrás finalizar el programa sin necesidad de reiniciar, perdiendo la información que no hubieras guardado.

Por ello te recomendamos como medida de prudencia que antes de ejecutar cualquier programa, y en especial desarrollos extensos, procedas a guardar el proyecto de programa a través de la opción Guardar de Visual Basic, o bien que transfieras el código a un procesador de textos y procedas a guardarlo como texto. De esta manera, ante bloqueos inesperados, podrás recuperar el trabajo realizado y proceder a corregirlo.

Existen diversos tipos de errores interceptables por Visual Basic.

Puedes consultar una lista de ellos en la ayuda del programa buscando "Errores interceptables". De esta lista, compuesta por más de 60 tipos de error, podemos extraer información relativa a los tipos de errores detectables, es decir, aquellos cuya ocurrencia de alguna manera podemos tratar a través de instrumentos de gestión de errores (On Error, objeto Err...).

Tratar de estudiar todos los tipos de error no tiene demasiado sentido. Más bien se tratará de ir familiarizándonos con ellos a través de nuestra interacción con el Visual Basic. Así pues, cuando se nos presente un error del tipo: "No hay suficiente espacio de pila", podemos ver a través de la ayuda cuáles son las causas de este error que afecta a la pila (espacio de memoria temporal). Por ejemplo que nuestro programa cuente con procedimientos anidados muy profundamente, cuestión frecuente con los procedimientos recursivos (es decir, los procedimientos que se llaman a sí mismos).

En cuestión de errores se trata de prevenir problemas pero también de buscar soluciones. Ya lo dice el refrán: "Burro cargado busca camino", o aplicado al ámbito de la programación "Programador con problemas busca soluciones".

Próxima entrega: CU00353A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

CAPTURA Y GESTIÓN DE ERRORES. INSTRUCCIÓN ON ERROR. OBJETO ERR. MÉTODO CLEAR.

Introduciremos la gestión de errores o protocolo para el tratamiento de los errores detectados durante la ejecución del programa a través de la instrucción **On Error**.

On Error activa una rutina de control de errores y especifica la ubicación de la misma. Vamos a ver algunas de las posibles sintaxis:

a) `On Error GoTo [Línea]`

donde la línea puede ser especificada como un número o como una etiqueta.

b) `On Error Resume Next`

c) `On Error GoTo 0`

En definitiva se trata de indicarle al programa qué hacer en caso de que se produzca un error durante la ejecución (recuerda que los errores en compilación van a impedir que se ejecute el programa).

En el caso a) hacemos que el programa salte a la línea especificada, que tendrá que estar en el mismo procedimiento en que se ha producido el error.

En el caso b) Resume Next indica que el programa salta a la siguiente instrucción después de la que ha generado el error (se intenta continuar con la ejecución ignorando el error).

Por último c) desactiva el control de errores, con lo que en caso de producirse uno el programa se detendrá o será impredecible.

Cuando se produce un error en tiempo de ejecución (error durante la ejecución del código), las propiedades del objeto Err se llenan con información que identifica al error de forma única. Antes del error o después de ser tratado las propiedades son las de defecto (en general cero, cadenas vacías o mensajes estándar). Las propiedades del objeto Err son:

a) **Number:** número de error. Vale cero (falso) si no se ha producido un error.
b) **Source:** nombre del programa (proyecto) en que se generó el error.
c) **Description:** texto informativo sobre el error.
d) **Helpfile:** unidad, ruta y nombre del archivo de ayuda de Visual Basic, si existe. En caso de no existir contiene una cadena vacía o espacio en blanco o cero.
e) **Helpcontext:** identificador de la parte de la ayuda asociada al número de error, si existe. En caso de no existir contiene una cadena vacía, espacio en blanco o cero.

© 2006-2029 aprenderaprogramar.com

Por ejemplo podemos hacer que sobre un Label se muestre la información Err.Description, que devuelve un mensaje informativo sobre el error. Por ejemplo "Nombre o número de archivo incorrecto", “La operación aritmética ha provocado un desbordamiento”, etc.

Prueba el siguiente código:

Código versiones menos recientes VB:

```
‘Curso Visual Basic aprenderaprogramar.com
Option Explicit

Private Sub Form_Load()
Dim i As Integer
Label1 = ""
On Error GoTo 88    '[Si ponemos On Error Resume Next sí se mostraría el texto]
i = Rnd * 10 ^ 16    '[Esta línea genera el error, i demasiado grande]
Label1 = Label1 & "Esta instrucción no se llega a ejecutar"
88 If Err Then MsgBox ("Se ha producido un error. Tipo de error = " & _
Err & " Descripción: " & Err.Description)
Label1 = Label1 & "La ejecución continúa"
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim i As Integer
        Label1.Text = ""
        On Error GoTo 88    '[Si ponemos On Error Resume Next sí se mostraría el
texto en el label]
        i = Rnd() * 10 ^ 16    '[Esta línea genera el error, i demasiado grande]
        Label1.Text = ("Esta instrucción no se llega a ejecutar")

88:     If Err.Number Then MsgBox("Se ha producido un error. Tipo de error = " & _
           Err.Number & " Descripción: " & Err.Description)
        Label1.Text = ("La ejecución continúa")
    End Sub
End Class
```

Con este programa generamos un error de desbordamiento (i excede los valores admisibles para Integer) y nos aparece un MsgBox que dice: “Se ha producido un error. Tipo de error = 6. Descripción: Desbordamiento” o “Descripción: la operación aritmética ha provocado un desbordamiento”.

Tras el tratamiento Err vuelve a establecerse a valores de defecto.

Podemos borrar los contenidos del objeto Err haciendo uso de la sintaxis Err.Clear (en las versiones más recientes Err.Clear()). A través de Clear se borran todas las propiedades establecidas del objeto Err. La invocación de Clear puede ser hecha por el programador, o bien ser automática tras ejecutarse un Resume, Exit Sub, Exit Function, Exit Property ó una instrucción On Error.

© 2006-2029 aprenderaprogramar.com

En el programa anterior, si después de la línea 88 mostramos Err.Number sobre un Label nos devuelve un 6. En cambio, si escribimos Err.Clear y luego mostramos Err.Number sobre un Label, nos devuelve un cero, pues hemos restablecido Err a sus valores iniciales.

Prueba este otro código:

Código versiones menos recientes VB:

```
'Curso Visual Basic aprenderaprogramar.com
Option Explicit

Private Sub Form_Load()
On Error GoTo Gestionaerror
Dim i As Integer
i = Rnd * 10 ^ 16   '[Esta línea genera el error]
Label1 = "La ejecución continúa aquí debido al Resume Next" & vbCrLf
Label1 = Label1 & i   '[Devuelve cero ya que fue imposible asignarle valor tipo integer]
Gestionaerror:
If Err.Number <> 0 Then
   GestiónError
   Resume Next
End If
End Sub

Private Sub GestiónError()
MsgBox("Se ha producido un error. Tipo de error = " & Err.Number & ". Descripción: " & Err.Description)
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        On Error GoTo Gestionaerror
        Dim i As Integer
        i = Rnd() * 10 ^ 16    '[Esta línea genera el error]
        Label1.Text = "La ejecución continúa aquí debido al Resume Next" & vbCrLf
        Label1.Text = Label1.Text & i    '[Devuelve cero ya que fue imposible asignarle valor
tipo integer]
Gestionaerror:
        If Err.Number <> 0 Then
            GestiónError()
            Resume Next
        End If
    End Sub

    Private Sub GestiónError()
    MsgBox("Se ha producido un error. Tipo de error = " & Err.Number & ". Descripción: " &
Err.Description)
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

En esta ocasión tenemos una etiqueta de línea a donde se remite el flujo en caso de error, y a través de esta línea se redirecciona nuevamente el flujo a un procedimiento denominado GestiónError. Ten en cuenta que al usar Resume Next se continúa la ejecución en la instrucción inmediatamente posterior a la que originó el error, y que también ocurre que las propiedades del objeto Err se restablecen. En caso contrario, se volvería a entrar en GestiónError una segunda vez.

Visual Basic dispone de más instrucciones y posibilidades relacionadas con la captura y gestión de errores, que es una parte importante cuando se desarrollan programas a nivel profesional. Nosotros en este curso nos limitamos a exponer algunas ideas básicas sobre qué son los errores y sobre las posibilidades para tratarlos.

Próxima entrega: CU00354A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

EJERCICIO

Se ha desarrollado un programa (programa R02) cuyo pseudocódigo se indica a continuación. Este programa consta de dos subprogramas llamados GeneraDato y Raíz. A través de GeneraDato se solicita un número entre 1 y 20 al usuario, mientras que Raíz devuelve la raíz cuadrada de ese número entero.

Se desea crear un programa R03 con Visual Basic que realice lo previsto para el programa R02 y que además introduzca un procedimiento para gestión de errores basado en On Error que impida la parada de este programa en caso de que el usuario no introduzca el tipo de dato correcto (por ejemplo que introduzca una letra en lugar de un número). Si el usuario introduce un dato incorrecto, el programa deberá recuperarse, mostrar un mensaje informativo indicando que se ha producido un error, mostrar que la raíz de cero es cero y terminar.

Nota: usaremos inputBox para solicitar datos en Visual Basic y msgBox para mostrar mensajes de aviso o de resultados.

```
'Curso VB aprenderaprogramar.com - Pseudocódigo
PROGRAMA R02

Variables
    Enteras: Dato

1. Inicio
    2. Llamar Generadato
    3. Llamar Raiz(Dato)
4. Fin

Módulo Generadato
    1. Mientras Dato < 1 ó Dato > 20
        Mostrar "Introduzca un número entero entre 1 y 20"
        Pedir Dato [Aquí puede originarse el error]
    Repetir
FinMódulo

Módulo Raiz(Numero: Enteros)
    Variables
        Reales: Resultado
    1. Resultado = SQR(Numero)
    2. Mostrar "La raíz de", Numero, "es", Resultado
FinMódulo
```

© 2006-2029 aprenderaprogramar.com

SOLUCIÓN

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
'[Programa R03]

Option Explicit
Dim Dato As Integer

Private Sub Form_Load()
On Error GoTo ErrorControl
Call Generadato
Call Raíz(Dato)
ErrorControl:
If Err.Number <> 0 Then
   Call GestiónError
   Resume Next
End If
End Sub

Private Sub Generadato()
Do While Dato < 1 Or Dato > 20
   Dato = InputBox("Introduzca un entero entre 1 y 20")
Loop
End Sub

Private Sub Raíz(Número As Integer)
Dim Resultado As Single
Resultado = Sqr(Número)
MsgBox("La raíz de " & Número & " es " & Resultado)
End Sub

Private Sub GestiónError()
MsgBox("Se ha producido un error. Es posible que los datos que usted proporciona no se ajusten a lo solicitado. Compruébelo por favor")
Dato = 0
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Dim Dato As Integer
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        On Error GoTo ErrorControl
        Call Generadato()
        Call Raíz(Dato)
ErrorControl:
        If Err.Number <> 0 Then
            Call GestiónError()
            Resume Next
        End If
    End Sub

Private Sub Generadato()
  Do While Dato < 1 Or Dato > 20
  Dato = InputBox("Introduzca un entero entre 1 y 20")
  Loop
End Sub

Private Sub Raíz(ByVal Número As Integer)
   Dim Resultado As Single
   Resultado = Math.Sqrt(Número)
   MsgBox("La raíz de " & Número & " es " & Resultado)
End Sub

Private Sub GestiónError()
        MsgBox("Se ha producido un error. Es posible
que los datos que usted proporciona no se ajusten a lo
solicitado. Compruébelo por favor")
        Dato = 0
End Sub
End Class
```

Próxima entrega: CU00355A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

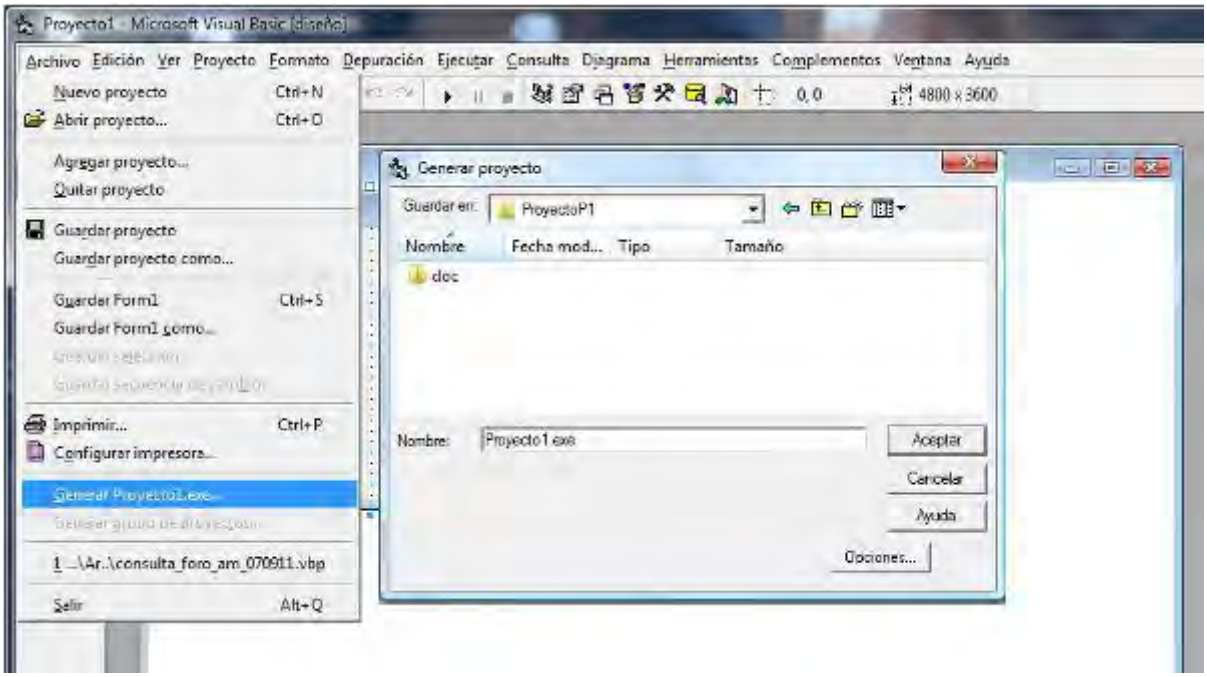
© 2006-2029 aprenderaprogramar.com

GENERAR UN ARCHIVO PUNTO EXE

Tras desarrollar el código correspondiente a un programa, el paso final será generar un archivo del tipo programa.exe que podamos ejecutar en cualquier ordenador con sistema operativo Windows sin necesidad de tener instalado Visual Basic.

Para ello seguiremos estos pasos:

a) **En las versiones menos recientes de Visual Basic:** ir al menú Archivo, Generar.exe, e indicar el nombre y la ruta donde queremos guardar nuestro archivo ejecutable. Pruébalo con cualquiera de los pequeños programas que hemos visto.



b) **En las versiones más recientes de Visual Basic:** como primer paso habremos creado un proyecto (para ello en el menú Archivo -- > Nuevo proyecto, elegimos Aplicación de Windows Forms). Escribimos el código de nuestro proyecto y guardamos el proyecto con un nombre, por ejemplo WindowsApplication1, en la ruta que nosotros queramos (por ejemplo C:\Users\Asus\Desktop\proyecto1). Podemos trabajar el tiempo que queramos en el proyecto, realizando los cambios o modificaciones que sean necesarios. Cuando queramos generar el archivo ejecutable (punto exe) vamos a la opción de menú Depurar -- > Generar WindowsApplication1 (o generar el proyecto con el nombre que le hayamos puesto). Una vez hecho esto en una ruta como C:\Users\Asus\Desktop\proyecto1\WindowsApplication1 \WindowsApplication1\bin\Release encontraremos un archivo con extensión .exe y el nombre de nuestra aplicación (si no lo encuentras búscalo con el buscador del explorador de Windows). Por ejemplo WindowsApplication1.exe. Haciendo doble click sobre ese archivo podemos ejecutar nuestro programa. También podemos copiarlo y pegarlo en otra ubicación (o trasladarlo a otro computador).

© 2006-2029 aprenderaprogramar.com

Una vez tenemos el archivo punto exe, este archivo funcionará como cualquier otro programa. Podemos distribuirlo a cualquier usuario de Windows y éste podrá ejecutarlo en su ordenador, simplemente haciendo doble click sobre él. No obstante, al distribuirlo como punto exe el código de programación que constituye el programa deja de estar accesible, de forma que un usuario que tenga el punto exe no puede acceder al código del programa, sino simplemente ejecutarlo.

FINAL DEL CURSO PROGRAMACIÓN EN VISUAL BASIC NIVEL I

Con esta entrega llegamos al final del curso "Programación en Visual Basic nivel I". Esperamos que haya sido un curso útil y ameno para todas las personas que lo hayan seguido. Y como en todo final, cabe hacer algunas consideraciones especiales.

- Gracias al equipo humano de aprenderaprogramar.com que ha hecho posible su publicación, y en especial a César Hernández, Javier Roa, Jorge Maestro, Manuel Tello, Walter Sagástegui y Enrique González.

- Gracias a todas las personas que de una u otra forma han participado enviando propuestas de mejora, comentarios, avisos de erratas, etc. y a los alumnos que han seguido el curso en las modalidades de tickets on-line y tutorizado on-line.

- A todos los que no han participado pero han seguido el curso, desde aprenderaprogramar.com les agradeceríamos nos hicieran llegar una opinión o propuesta de mejora sobre el mismo, bien a través de correo electrónico a contacto@aprenderaprogramar.com, bien a través de los foros.

- A quienes hayan seguido el curso de forma gratuita y piensen que los contenidos son de calidad y que merece dar un pequeño apoyo económico para que se puedan seguir ofreciendo más y mejores contenidos en este sitio web, les estaremos muy agradecidos si realizan una pequeña aportación económica en forma de donación pulsando sobre el enlace que aparece en la página principal de aprenderaprogramar.com.

Como colofón al curso, vamos a proponer un ejercicio complementario y opcional para aquellas personas que estén interesadas en realizarlo. Este ejercicio es un complemento al curso, con objeto de que las personas interesadas puedan practicar con una aplicación algorítmica desarrollada en código Visual Basic. Este ejercicio puede requerir varias horas de dedicación, por lo que insistimos en que es opcional ya que no trata de mostrar conceptos básicos, sino de hacer una aplicación práctica semiavanzada. Diremos también que es interesante, pero cada cual que decida en función de sus intereses y tiempo disponible.

A todos los que nos han leído y nos siguen, gracias. ¡Nos vemos en el próximo!

El equipo de aprenderaprogramar.com

© 2006-2029 aprenderaprogramar.com

Próxima entrega: CU00356A (opcional)

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

Ejercicio complementario y opcional para el curso Programación en Visual Basic. Nivel I.

EJERCICIO DE ALGORITMIA RESUELTO CON VISUAL BASIC: ORDENAR SERIES DE SERIES DE NÚMEROS

El ejercicio consiste en generar un programa enfocado al programador (un programa para verificar el funcionamiento del algoritmo y no para realizar un proceso orientado al usuario) capaz de ordenar una serie de series de números de mayor a menor.

El objetivo del programa es realizar un proceso análogo al de ordenar palabras por orden alfabético, pero en este caso usando números. Consideraremos que una serie de números es mayor que otra cuando un elemento de esa serie en igual posición que otros es mayor que ellos dentro del orden de prelación definido por las columnas. En caso de empate entre dos series, el desempate se dirime evaluando el siguiente elemento de la serie sólo entre las series empatadas.

Ejemplo 1: Serie 1: 100, 45, 33, 2500

Serie 2: 111, 22, 112, 335

Serie 3: 44, 22, 1, 3

Las series se ordenarían así: serie 2 > serie 1 > serie 3 , en base a que tras comparar los elementos nº1 de cada serie, ha resultado ese orden y no existen empates.

Ejemplo 2: Serie 1: 5500, 35, 23, 6500

Serie 2: 5500, 22, 11, 3345

Serie 3: 43500, 33, 2, 2

Las series se ordenarían así: serie 3 > serie 1 > serie 2, en base a que tras comparar los elementos nº1 resulta que es la serie 3 la primera y queda como empate la segunda y tercera series. Al existir empate en un elemento, se pasa al siguiente pero sólo entre aquellas series que han resultado empatadas. Del análisis del elemento nº2 resulta que es mayor en la serie 1 que en la serie 2, con lo cual la serie 1 pasa a ocupar el orden segundo y la serie 2 el orden tercero.

SOLUCIÓN

El trasfondo de este programa gira en torno a la siguiente pregunta: ¿Cómo podríamos ordenar palabras si el lenguaje informático con el que trabajamos no fuera capaz de comparar dos palabras? Si empiezas a manejar cadenas (palabras) con Visual Basic comprobarás que ordenar palabras es relativamente sencillo gracias a que el lenguaje es capaz de comparar dos palabras y decir cuál es "mayor" y cuál es "menor" (orden alfabético). Si el lenguaje no fuera capaz de esto, probablemente recurriríamos a tratar de transformar las letras en números, quedándonos con un problema de ordenar números de solución aparentemente trivial.

© 2006-2029 aprenderaprogramar.com

Hasta ahí parece todo claro. La variante que se nos presenta es que los problemas de ordenar números que hemos tratado hasta ahora consisten en **ordenar una serie** de números mientras que ahora queremos **ordenar series** de números. La dificultad radica en que es necesario ir comparando más o menos elementos y series en función de la existencia o no de empates.

La primera cuestión a valorar es que el problema goza de cierta complejidad y por tanto, como siempre pero con más razón si cabe, debemos dedicar tiempo y esfuerzo a su planteamiento y resolución sobre el papel antes de ponernos a programar. En caso contrario (puedes probarlo), con mucha probabilidad nos veremos atrapados en un maremagnum de código con el que creemos estar cerca de la solución a la que sin embargo no llegamos. Y así pasan horas... y días... y si te descuidas semanas.

Una cuestión a valorar dentro de los aspectos previos es: **¿El problema se puede resolver a través de un algoritmo?**

Como paso preliminarísimo nos podemos preguntar si es posible definir un algoritmo para resolver el problema. Para responder a esto, pensemos en cómo ordenamos palabras los humanos... Cojamos un lápiz y un papel y escribamos una lista de palabras. Procedamos a ordenarlas, marcando junto a cada una de ellas el orden que ocupan en la lista (1, 2 , 3...) ¿Estaremos siguiendo un proceso lógico y ordenado? La respuesta es sí, con lo cual "podemos programarlo". ¿Seguimos siempre el mismo proceso los humanos? La respuesta es que quizás no, porque quizás no aplicamos la misma técnica a ordenar dos palabras que a ordenar 200. Si tenemos una lista de 2 palabras, quizás analizamos cuál es la primera y asignamos el orden segundo a la que queda. En cambio, con una lista de 200, podemos analizar cuál es la primera... y empezar a tener problemas (por ejemplo, 24 palabras que tengan las primeras 3 letras iniciales iguales). A partir de ahí, empezamos a utilizar técnicas como el marcado de palabras y letras (mental o sobre el papel), asignación de órdenes relativos entre palabras empatadas y asignación de órdenes absolutos dentro de la lista, eliminación de palabras con orden determinado, etc.

Bien, "creemos" que haya algún algoritmo que nos permita ordenar cualquier lista de números o palabras, puesto que los humanos lo hacemos. Vamos a tratar de centrar el problema con las pautas expuestas en el curso “Bases de la programación nivel I” de aprenderaprogramar.com:

1. ¿Cuál es mi objetivo?
2. ¿Cuáles son los condicionantes que afectan al problema?
3. ¿Qué método o esquema de resolución voy a aplicar?
4. ¿Cuáles son los datos de partida?
5. ¿Qué resultados quiero obtener?

1. ¿Cuál es mi objetivo?

Desarrollar un programa capaz de ordenar una serie de series de números.

© 2006-2029 aprenderaprogramar.com

2. ¿Cuáles son los condicionantes que afectan al problema?

Entre los posibles condicionantes que se nos ocurren como reflexión inicial tendríamos:

a) Cantidad de series.
b) Longitud de las series: ¿las series tienen todas igual cantidad de números? ¿o unas tienen más números que otras?
c) Los números ¿son naturales?, ¿son enteros?, ¿son enteros excluido el cero...?
d) Existencia de empates y su localización: ¿Existen empates en el primer número de la serie? ¿Sólo en el primer número y no en los siguientes? ¿No existen empates? ¿Existen pocos empates? ¿Existen muchos empates?

El planteamiento inicial con el que vamos a trabajar es el de orientación "genérica" pero a partir de:

- Cualquier número de series.
- Series de igual longitud.
- Números enteros positivos.
- Cualquier posibilidad de casos en relación a empates: inexistentes, escasos o abundantes.

3. ¿Qué método o esquema de resolución voy a aplicar?

Desarrollaremos este punto de forma extensa posteriormente.

4. ¿Cuáles son los datos de partida?

Dado que se trata de un programa de autodesarrollo o programa-herramienta para su uso en otros programas, vamos a plantear la siguiente estrategia:

- El número de filas y de columnas serán establecidos por el programador en el propio código.
- Los datos serán generados de forma aleatoria (por supuesto, controlada) o bien serán definidos por el programador a través de código del programa.

5. ¿Qué resultados quiero obtener?

Nos vamos a limitar a obtener una pantalla donde el ordenador nos dice cuál es la matriz a ordenar (cuya generación controlaremos a través del código) y cuál es la matriz ordenada (resultado del algoritmo de ordenación que planteemos).

MÉTODO O ESQUEMA DE RESOLUCIÓN

Documentarnos

¿Cómo enfrentarnos al problema? Quizás hay distintas vías de resolución (igual que hay distintas formas para ordenar una serie de números) pero el primer paso, y será muy difícil empezar a hacer algo sin darlo, será el de documentarnos y plantear casos de series de números a ordenar para tratar de determinar una metodología común para resolverlos todos ellos. ¿Qué metodología de resolución usaremos? Podríamos pensar en ordenar los elementos de cada columna. Sin embargo, viendo unos

© 2006-2029 aprenderaprogramar.com

cuantos casos podemos llegar a la conclusión de que la matriz ordenada por columnas no nos permite resolver el problema, porque nos dice cuál es el mayor valor de cada columna pero no a qué serie pertenece ese valor ni si ese valor tiene importancia para determinar el orden de la serie dentro de la lista. Por tanto no basta con ordenar series de números.

Parece claro que vamos a necesitar "marcadores", o asociar a un valor por ejemplo la serie a que pertenece. ¿Cuántos marcadores utilizar? Seguramente se nos pueden ocurrir distintas alternativas. No obstante, casi el problema se restringe a dos opciones:

a) Desarrollar un algoritmo de ordenación.
b) Tratar de emular el procedimiento humano o "algoritmo humano".

Si buscáramos desarrollar un algoritmo "óptimo" podríamos optar por buscar información sobre métodos de ordenación de este tipo en libros, internet, revistas, etc. En nuestro caso el objetivo es didáctico y nos vamos a decantar por emular el proceder humano. Los motivos para ello:

1. Sabemos que funciona
2. Sabemos aplicarlo, aunque no tengamos "el código".
3. Probablemente nos resulte más sencillo trabajar partiendo de algo que ya sabemos que tratar de determinar nuevos métodos.
4. Todo ello nos lleva a que sea didáctico, uno de los fines de este curso.

Bien, vamos a emular el proceder humano y, ¿cómo lo hacemos?

A cada elemento de la matriz le podemos asignar un valor, el orden que ocupa dentro de la columna a la que pertenece y el orden de su serie dentro de la lista.

Elemento "origen"	Conceptos asociados
Elemento de la matriz que queremos ordenar	Valor numérico Orden dentro de su columna Orden de su serie

Una forma de plasmar este planteamiento con Visual Basic se basa en crear un tipo de datos que nos permita mantener toda esta información agrupada. En las versiones menos recientes de Visual Basic esto se hace con lo que se denomina tipos definidos por el usuario o variables UDT (User Defined Type), y en las versiones más recientes se denomina estructuras o Structures. No hemos estudiado UDT ni Structures en el curso pero vamos a usar este tipo de datos en este ejercicio porque resultan de utilidad, lo cual no significa que sean "imprescindibles". La creación de este tipo de datos se hace así:

Versiones menos recientes de VB:

```
Private Type element
      Valor As Single
      Ordenensucolumna As Integer
      Ordendesuserie As Integer
End Type
```

Versiones más recientes de VB:

```
Private Structure element
    Dim Valor As Single
    Dim Ordenensucolumna As Integer
    Dim Ordendesuserie As Integer
End Structure
```

© 2006-2029 aprenderaprogramar.com

El proceso humano, aunque puede tener sus variantes, sería algo similar a lo siguiente:

1. Análisis de columna y establecer órdenes relativos y empates.

2. Si se desprende un orden definitivo, asignarlo, eliminar la serie del proceso de evaluación y volver a comenzar el proceso de evaluación para todos los elementos restantes en la primera columna (esto es discutible pero lo haremos así de momento).

3. **En las palabras empatadas (sólo las empatadas)**, aplicar el punto 1 en la siguiente columna. Si persiste un empate, repetir el punto 3. Si se termina el proceso y persiste un empate es que dos o más series son iguales. En este caso, se les daría un número de orden indistinto, ya que son iguales, y se eliminarían del proceso de evaluación. Se continuaría volviendo a comenzar el proceso de evaluación para todos los elementos restantes en la primera columna.

En la siguiente entrega iremos viendo cómo terminar de plantear el problema y transformarlo en código.

Próxima entrega: CU00357A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

Ejercicio complementario y opcional para el curso Programación en Visual Basic. Nivel I.

EJERCICIO RESUELTO CON VISUAL BASIC. PLANTEAR CASOS, RELACIONES Y PROCESOS CON LETRAS

Plantear casos. Método. Valoraciones preliminares. Relaciones y procesos con letras.

Hemos de programar pensando en todas las situaciones posibles (al menos todas las que se nos ocurran), de forma que el programa sea capaz de resolverlas todas. El número de casos y el número de series por caso lo elegimos en base a nuestra experiencia en resolución de programas, a nuestra intuición o creencia de que estamos planteando casos particulares que nos permitirán la resolución y por último, como respuesta al ensayo y error. Es decir, si planteamos dos casos y después vemos que el programa no responde bien ante determinadas circunstancias, tendremos que replantear usando distinto número o tipo de casos. En base a lo expuesto plantearemos distintas situaciones usando cuatro series como:

Caso 1. Una serie mayor no empatada y las tres restantes empatadas.
Caso 2. Las series mayores con triple empate en su primer número y sin empate en su segundo número.
Caso 3. Las series mayores con triple empate en su primer número, las dos primeras con empate en su segundo número.
Caso 4. Una serie mayor no empatada y las dos siguientes con empate absoluto en sus cuatro números.

Para cada caso escribiremos el proceso primero con un lenguaje más libre (hágalo cada cual como le parezca, a modo de tanteo y reflexión inicial) y después tratando de aproximarnos a un metalenguaje lógico-matemático.

El planteamiento de casos y el metalenguaje previo al pseudocódigo sería el siguiente:

Caso 1

Paso 1: $\begin{pmatrix} 18 & 55 & 32 & 41 \\ 22 & 33 & 85 & 57 \\ 18 & 51 & 82 & 33 \\ 18 & 55 & 19 & 11 \end{pmatrix}$ Es la matriz a analizar (ok tiene más de una fila).

Paso 2: Extracción de la columna 1 $\begin{pmatrix} 18 \\ 22 \\ 18 \\ 18 \end{pmatrix}$

© 2006-2029 aprenderaprogramar.com

Paso 3: ¿Orden $\begin{pmatrix} 18 \\ 22 \\ 18 \\ 18 \end{pmatrix}$?

Paso 4: 22 > 18

Paso 5: Orden ① = 22 33 85 57

Paso 6: Nueva matriz a analizar (ok tiene más de una fila) $\begin{pmatrix} 18 & 55 & 32 & 41 \\ 18 & 51 & 82 & 33 \\ 18 & 55 & 19 & 11 \end{pmatrix}$

Paso 7: Extracción de la columna 1 $\begin{pmatrix} 18 \\ 18 \\ 18 \end{pmatrix}$

Paso 8: ¿Orden $\begin{pmatrix} 18 \\ 18 \\ 18 \end{pmatrix}$?

Paso 9: 18 = 18 = 18 Triple empate

Paso 10: Sub – matriz a analizar $\begin{pmatrix} 55 & 32 & 41 \\ 51 & 82 & 33 \\ 55 & 19 & 11 \end{pmatrix}$

Paso 11: Extracción de la columna 1 $\begin{pmatrix} 55 \\ 51 \\ 55 \end{pmatrix}$

Paso 12: ¿Orden $\begin{pmatrix} 55 \\ 51 \\ 55 \end{pmatrix}$?

Paso 13: 55 = 55 > 51 doble empate

Paso 14: Sub – matriz a analizar $\begin{pmatrix} 32 & 41 \\ 19 & 11 \end{pmatrix}$

Paso 15: Extracción de la columna 1 $\begin{pmatrix} 32 \\ 19 \end{pmatrix}$

Paso 16: ¿Orden $\begin{pmatrix} 32 \\ 19 \end{pmatrix}$?

Paso 17: 32 > 19

© 2006-2029 aprenderaprogramar.com

Paso 18: Orden ② = 18 55 32 41

Paso 19: Nueva matriz a analizar (ok tiene más de una fila) $\begin{pmatrix} 18 & 51 & 82 & 33 \\ 18 & 55 & 19 & 11 \end{pmatrix}$

Paso 20: Extracción de la columna 1 $\begin{pmatrix} 18 \\ 18 \end{pmatrix}$

Paso 21: ¿Orden $\begin{pmatrix} 18 \\ 18 \end{pmatrix}$?

Paso 22: 18 = 18 Doble empate

Paso 23: Sub – matriz a analizar $\begin{pmatrix} 51 & 82 & 33 \\ 55 & 19 & 11 \end{pmatrix}$

Paso 24: Extracción de la columna 1 $\begin{pmatrix} 51 \\ 55 \end{pmatrix}$

Paso 25: 55 > 51

Paso 26: Orden ③ = 18 55 19 11

Paso 27: Nueva matriz a analizar $\begin{pmatrix} 18 & 51 & 82 & 33 \end{pmatrix}$ Sólo tiene una fila

Paso 28: Orden ④ = 18 51 82 33

Caso 2

Paso 1: $\begin{pmatrix} 33 & 21 & 13 & 10 \\ 12 & 17 & 31 & 55 \\ 33 & 73 & 85 & 15 \\ 33 & 22 & 25 & 77 \end{pmatrix}$ Es la matriz a analizar (ok tiene más de una fila).

Paso 2: Extracción de la columna 1 $\begin{pmatrix} 33 \\ 12 \\ 33 \\ 33 \end{pmatrix}$

Paso 3: ¿Orden $\begin{pmatrix} 33 \\ 12 \\ 33 \\ 33 \end{pmatrix}$?

Paso 4: 33 = 33 = 33 > 12 Triple empate

© 2006-2029 aprenderaprogramar.com

Paso 5: Sub – matriz a analizar $\begin{pmatrix} 21 & 13 & 10 \\ 73 & 85 & 15 \\ 22 & 25 & 77 \end{pmatrix}$

Paso 6: Extracción de la columna 1 $\begin{pmatrix} 21 \\ 73 \\ 22 \end{pmatrix}$

Paso 7: ¿Orden $\begin{pmatrix} 21 \\ 73 \\ 22 \end{pmatrix}$?

Paso 8: 73 > 22

Paso 9: Orden ① = 33 73 85 15

Paso 10: Nueva matriz a analizar (ok tiene más de una fila) $\begin{pmatrix} 33 & 21 & 13 & 10 \\ 12 & 17 & 31 & 55 \\ 33 & 22 & 25 & 77 \end{pmatrix}$

Paso 11: Extracción de la columna 1 $\begin{pmatrix} 33 \\ 12 \\ 33 \end{pmatrix}$

Paso 12: ¿Orden $\begin{pmatrix} 33 \\ 12 \\ 33 \end{pmatrix}$?

Paso 13: 33 = 33 > 12 Doble empate

Paso 14: Sub – matriz a analizar $\begin{pmatrix} 21 & 13 & 10 \\ 22 & 25 & 77 \end{pmatrix}$

Paso 15: Extracción de la columna 1 $\begin{pmatrix} 21 \\ 22 \end{pmatrix}$

Paso 16: ¿Orden $\begin{pmatrix} 21 \\ 22 \end{pmatrix}$?

Paso 17: 22 > 21

Paso 18: Orden ② = 33 22 25 77

Paso 19: Nueva matriz a analizar (ok tiene de una fila) $\begin{pmatrix} 33 & 21 & 13 & 10 \\ 12 & 17 & 31 & 55 \end{pmatrix}$

Paso 20: Extracción de la columna 1 $\begin{pmatrix} 33 \\ 12 \end{pmatrix}$

© 2006-2029 aprenderaprogramar.com

Paso 21: ¿Orden $\begin{pmatrix} 33 \\ 12 \end{pmatrix}$?

Paso 22: 33 > 12

Paso 23: Orden ③ = 33 21 13 10

Paso 24: Nueva matriz a analizar $\begin{pmatrix} 12 & 17 & 31 & 55 \end{pmatrix}$ Sólo tiene una fila.

Paso 25: Orden ④ = 12 17 31 55

Caso 3

Paso 1: $\begin{pmatrix} 56 & 22 & 17 & 81 \\ 31 & 12 & 14 & 16 \\ 56 & 22 & 30 & 60 \\ 56 & 10 & 20 & 32 \end{pmatrix}$ Es la matriz a analizar (ok tiene más de una fila).

Paso 2: Extracción de la columna 1 $\begin{pmatrix} 56 \\ 31 \\ 56 \\ 56 \end{pmatrix}$

Paso 3: ¿Orden $\begin{pmatrix} 56 \\ 31 \\ 56 \\ 56 \end{pmatrix}$?

Paso 4: 56 = 56 = 56 > 31 Triple empate

Paso 5: Sub – matriz a analizar $\begin{pmatrix} 22 & 17 & 81 \\ 22 & 30 & 60 \\ 10 & 20 & 32 \end{pmatrix}$

Paso 6: Extracción de la columna 1 $\begin{pmatrix} 22 \\ 22 \\ 10 \end{pmatrix}$

Paso 7: ¿Orden $\begin{pmatrix} 22 \\ 22 \\ 10 \end{pmatrix}$?

Paso 8: 22 = 22 > 10 Doble empate

Paso 9: Sub – matriz a analizar $\begin{pmatrix} 17 & 81 \\ 30 & 60 \end{pmatrix}$

© 2006-2029 aprenderaprogramar.com

Paso 10: Extracción de la columna 1 $\begin{pmatrix} 17 \\ 30 \end{pmatrix}$

Paso 11: ¿Orden $\begin{pmatrix} 17 \\ 30 \end{pmatrix}$?

Paso 12: 30 > 17

Paso 13: Orden ① = 56 22 30 60

Paso 14: Nueva matriz a analizar (ok tiene más de una fila) $\begin{pmatrix} 56 & 22 & 17 & 81 \\ 31 & 12 & 14 & 16 \\ 56 & 10 & 20 & 32 \end{pmatrix}$

Paso 15: Extracción de la columna 1 $\begin{pmatrix} 56 \\ 31 \\ 56 \end{pmatrix}$

Paso 16: ¿Orden $\begin{pmatrix} 56 \\ 31 \\ 56 \end{pmatrix}$?

Paso 17: 56 = 56 > 31 Doble empate

Paso 18: Sub – matriz a analizar $\begin{pmatrix} 22 & 17 & 81 \\ 10 & 20 & 32 \end{pmatrix}$

Paso 19: Extracción de la columna 1 $\begin{pmatrix} 22 \\ 10 \end{pmatrix}$

Paso 20: ¿Orden $\begin{pmatrix} 22 \\ 10 \end{pmatrix}$?

Paso 21: 22 > 10

Paso 22: Orden ② = 56 22 17 81

Paso 23: Nueva matriz a analizar (ok tiene más de una fila) $\begin{pmatrix} 31 & 12 & 14 & 16 \\ 56 & 10 & 20 & 32 \end{pmatrix}$

Paso 24: Extracción de la columna 1 $\begin{pmatrix} 31 \\ 56 \end{pmatrix}$

Paso 25: ¿Orden $\begin{pmatrix} 31 \\ 56 \end{pmatrix}$?

Paso 26: 56 > 31

Paso 27: Orden ③ = 56 10 20 32

Paso 28: Nueva matriz a analizar $\begin{pmatrix} 31 & 12 & 14 & 16 \end{pmatrix}$ Sólo tiene una fila.

Paso 29: Orden ④ = 31 12 14 16

© 2006-2029 aprenderaprogramar.com

Caso 4

Paso 1: $\begin{pmatrix} 73 & 18 & 21 & 43 \\ 85 & 91 & 17 & 12 \\ 73 & 18 & 21 & 43 \\ 66 & 12 & 17 & 13 \end{pmatrix}$ Es la matriz a analizar (ok tiene más de una fila).

Paso 2: Extracción de la columna 1 $\begin{pmatrix} 73 \\ 85 \\ 73 \\ 66 \end{pmatrix}$

Paso 3: ¿Orden $\begin{pmatrix} 73 \\ 85 \\ 73 \\ 66 \end{pmatrix}$?

Paso 4: 85 > 73

Paso 5: Orden ① = 85 91 17 12

Paso 6: Nueva matriz a analizar (ok tiene más de una fila) $\begin{pmatrix} 73 & 18 & 21 & 43 \\ 73 & 18 & 21 & 43 \\ 66 & 12 & 17 & 13 \end{pmatrix}$

Paso 7: Extracción de la columna 1 $\begin{pmatrix} 73 \\ 73 \\ 66 \end{pmatrix}$

Paso 8: ¿Orden $\begin{pmatrix} 73 \\ 73 \\ 66 \end{pmatrix}$?

Paso 9: 73 = 73 > 66 Doble empate

Paso 10: Sub – matriz a analizar $\begin{pmatrix} 18 & 21 & 43 \\ 18 & 21 & 43 \end{pmatrix}$

Paso 11: Extracción de la columna 1 $\begin{pmatrix} 18 \\ 18 \end{pmatrix}$

Paso 12: ¿Orden $\begin{pmatrix} 18 \\ 18 \end{pmatrix}$?

Paso 13: 18 = 18 Doble empate

© 2006-2029 aprenderaprogramar.com

Paso 14: Sub – matriz a analizar $\begin{pmatrix} 21 & 43 \\ 21 & 43 \end{pmatrix}$

Paso 15: Extracción de la columna 1 $\begin{pmatrix} 21 \\ 21 \end{pmatrix}$

Paso 16: ¿Orden $\begin{pmatrix} 21 \\ 21 \end{pmatrix}$?

Paso 17: 21 = 21 Doble empate

Paso 18: Sub – matriz a analizar $\begin{pmatrix} 43 \\ 43 \end{pmatrix}$

Paso 19: Extracción de la columna 1 $\begin{pmatrix} 43 \\ 43 \end{pmatrix}$

Paso 20: ¿Orden $\begin{pmatrix} 43 \\ 43 \end{pmatrix}$?

Paso 21: 43 = 43 Doble empate

Paso 22: Sub – matriz a analizar $\rightarrow$ NO HAY SUB – MATRIZ : asignamos orden.

Paso 23: Orden ② = 73 18 21 43

Paso 24: Nueva matriz a analizar (ok tiene más de una fila) $\begin{pmatrix} 73 & 18 & 21 & 43 \\ 66 & 12 & 17 & 13 \end{pmatrix}$

Paso 25: Extracción de la columna 1 $\begin{pmatrix} 73 \\ 66 \end{pmatrix}$

Paso 26: ¿Orden $\begin{pmatrix} 73 \\ 66 \end{pmatrix}$?

Paso 27: 73 > 66
Paso 28: Orden ③ = 73 18 21 43
Paso 29: Nueva matriz a analizar $\begin{pmatrix} 66 & 12 & 17 & 13 \end{pmatrix}$ Sólo tiene una fila.
Paso 30: Orden ④ = 66 12 17 13

Hemos planteado casos con números en vez de hacerlo con letras. ¿Por qué? Un motivo para ello es que necesitamos hacer comparaciones para determinar cuándo un elemento es mayor que otro, cosa que no podríamos hacer con letras a las que no hubiéramos asignado un valor. Se han elegido matrices tipo 4x4 porque se estima que una 2x2 no es representativa de un caso genérico y que una 3x3 puede presentar particularidades indeseables.

Próxima entrega: CU00358A

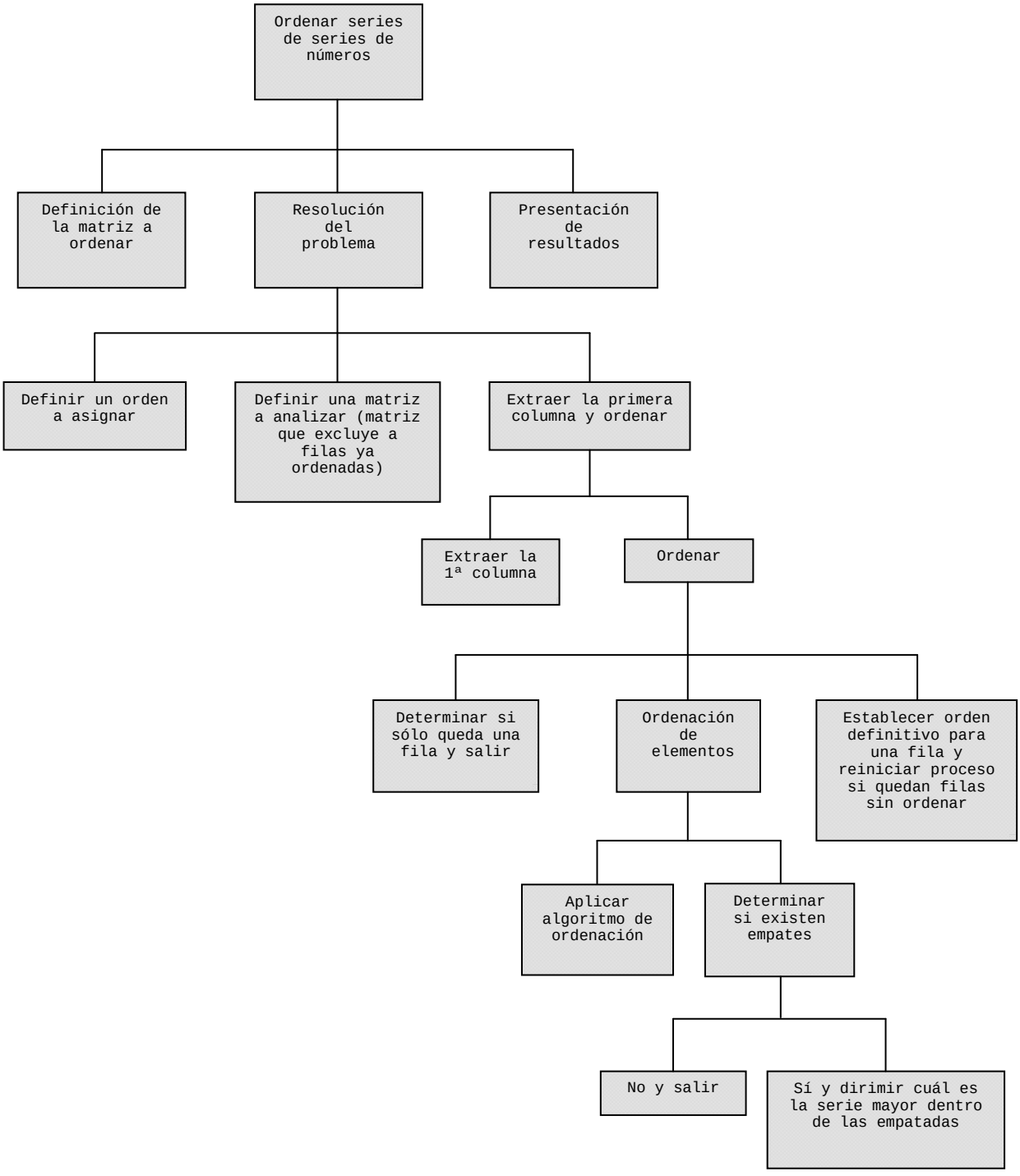
Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

Ejercicio complementario y opcional para el curso Programación en Visual Basic. Nivel I.

PLANTEAR EL PROCEDIMIENTO DE RESOLUCIÓN. ABSTRACCIÓN. ESQUEMAS DESCENDENTES.

En base a nuestra experiencia como programadores hemos hecho un planteamiento de casos y aplicado una metodología que estimamos se aproxima a un proceso tipo algoritmo, de momento con un lenguaje que no es pseudocódigo. Llega el momento de abstraer esos planteamientos concretos hacia un procedimiento genérico. Es un momento por tanto de reflexión respecto a cómo vamos a enfocar el programa. Analizando los casos anteriores podemos plantear el siguiente esquema descendente:



© 2006-2029 aprenderaprogramar.com

Este esquema representa una forma de enfocar la resolución del problema. Se entiende que desde un primer planteamiento hasta el esquema definitivo habrá una evolución derivada de la reflexión, mejoras del diseño, constatación de fallos, etc.

Ten en cuenta que si has hecho un correcto planteamiento de casos y organización de la estructura del programa, sólo es cuestión de tiempo llegar a un código satisfactorio. En cambio, si no has sido capaz de elegir casos adecuadamente, analizar los procesos unitarios asimilables a pasos para el ordenador, o si aún habiendo hecho lo anterior no eres capaz de aprehender la información de que dispones para conformar una estructura preliminar de programa, alcanzar un código satisfactorio puede ser un camino tortuoso o imposible.

PSEUDOCÓDIGO Y VERIFICACIONES PARCIALES. CONSTRUCCIÓN DE ABAJO A ARRIBA.

Llega el momento de plantear en pseudocódigo las expresiones o ideas abstractas que hemos ido desarrollando. No vamos a realizar el desarrollo de esta materia ya que resulta un proceso largo (consulta el curso Bases de la programación nivel I de aprenderaprogramar.com para más detalles). Se trata de que el programador plasme una estructura del programa y unos procesos en pseudocódigo para posteriormente ponerse delante del ordenador y de un lenguaje concretos.

¿Cómo empezar? Recuerda siempre que estamos trabajando con la metodología de "dividir y vencer". Por tanto, una buena forma de empezar es no mirar al problema en su conjunto, sino enfrentarnos a partes de él de forma independiente. Si lo estimamos necesario podemos plantear pseudocódigo para programas particulares y a partir de ellos construir un programa genérico.

Del esquema descendente anterior podemos sacar ideas de cuáles pueden ser partes independientes del programa (módulos), como definición de la matriz a ordenar, un algoritmo de ordenación para analizar las columnas que extraigamos, un proceso que permita eliminar empates, etc.

Cada programador puede llegar a distintas formas de solucionar el problema, existiendo métodos variantes por la forma de plantear y agrupar procesos, e incluso métodos distintos de mayor o menor eficiencia que otros.

CÓDIGO Y VERIFICACIONES PARA LA INTEGRACIÓN TOTAL Y ABSTRACCIÓN TOTAL.

Después de atacar el problema por partes que hemos considerado independientes y quizás con cierto grado de particularidad, tendremos que acometer la integración de las distintas partes en un todo, la generalización de las particularidades y la prueba del conjunto. El proceso puede ser relativamente directo o requerir aún de un trabajo importante.

Es de especial relevancia realizar una correcta verificación del programa que hayamos creado hasta constatar su total solidez. Si bien el número de casos utilizados en la búsqueda del método de resolución es limitado, el uso del ordenador nos permitirá hacer una verificación tan exhaustiva como deseemos.

Por motivos de espacio nos limitamos a ofrecer un código que podría ser el resultado al que llegamos después de recorrer el proceso de construcción que hemos omitido. Ten en cuenta que se utilizan alguna herramientas que no hemos abordado en el curso.

© 2006-2029 aprenderaprogramar.com

CÓDIGO DEL PROGRAMA ORNUMELP PARA VERSIONES MENOS RECIENTES DE VISUAL BASIC

'PROGRAMA ORNUMELP versión 0.1 aprenderaprogramar.com uso exclusivamente didáctico. Prohibido su uso comercial.

```
Option Explicit
Option Base 1
Dim a%, i%, j%, k%, t%, m%, Empates%
Dim TPO As Variant
Dim final As Boolean
'Fijamos el tamaño de la matriz declarando estas constantes
'Admite valores iguales o superiores a 2 (con una sola fila no hay orden que establecer)
Const numfilasmatoriginal = 4
Const numcolumnasmatoriginal = 4
Dim elementooriginal(1 To numfilasmatoriginal, 1 To numcolumnasmatoriginal)
Dim ordenaasignar%

Private Type element
valor As Single
serieaquepertenece As Integer
End Type

Dim elementoaanalizar() As element
Dim ordenencolumna() As element
Dim elementomatrizordenada() As element
Dim quématriz() As element
Dim subelemento() As element
Dim elementotemporal() As element

Private Sub Command1_Click()
'Nos permite ordenar la generación de una nueva matriz y su ordenamiento
final = False
ordenaasignar = 0
Label1 = " "
Form_Load
End Sub

Private Sub Form_Load()
Form1.Caption = "Ordenar series de números"
Asignarvaloresamatrizoriginal
Definirordenaasignar
Definirmatrizaanalizar
'Paso a extraer columna 1 y ordenar, indicándole qué matriz usar
ReDim quématriz(UBound(elementoaanalizar, 1), UBound(elementoaanalizar, 2))
For i = 1 To UBound(elementoaanalizar, 1)
  For j = 1 To UBound(elementoaanalizar, 2)
    quématriz(i, j).valor = elementoaanalizar(i, j).valor
    quématriz(i, j).serieaquepertenece = elementoaanalizar(i, j).serieaquepertenece
  Next j
Next i
Extraercolumna1yordenar
End Sub
Private Sub Asignarvaloresamatrizoriginal()
Show
'Asignación de valores directa por el programador, desactivada
'   elementooriginal(1, 1) = 73   elementooriginal(1, 2) = 18    elementooriginal(1, 3) = 21   elementooriginal(1, 4) = 43
'   elementooriginal(2, 1) = 85   elementooriginal(2, 2) = 91   elementooriginal(2, 3) = 17   elementooriginal(2, 4) = 12
'   elementooriginal(3, 1) = 73   elementooriginal(3, 2) = 18   elementooriginal(3, 3) = 21   elementooriginal(3, 4) = 43
'   elementooriginal(4, 1) = 66   elementooriginal(4, 2) = 12   elementooriginal(4, 3) = 17   elementooriginal(4, 4) = 13
'For i = 1 To numfilasmatoriginal
'   For j = 1 To numcolumnasmatoriginal
'     Label1 = Label1 & " " & elementooriginal(i, j)
'   Next j
'   Label1 = Label1 & Chr(13)
'Next i
ReDim elementomatrizordenada(numfilasmatoriginal, numcolumnasmatoriginal)
'Generación de una matriz de valores aleatorios; controlamos el número de empates
'con el múltiplo del RND; ACTIVADO
Label1 = "La matriz a ordenar es esta: " & Chr(13) & Chr(13)
For i = 1 To numfilasmatoriginal
  For j = 1 To numcolumnasmatoriginal
```

© 2006-2029 aprenderaprogramar.com

```
      Randomize
      elementooriginal(i, j) = Int(Rnd * 9) + 1
      Label1 = Label1 & elementooriginal(i, j) & " "
   Next j
   Label1 = Label1 & Chr(13)
Next i
End Sub

Private Sub Definirordenaasignar()
ordenaasignar = ordenaasignar + 1
End Sub

Private Sub Definirmatrizaanalizar()
Show
If ordenaasignar > numfilasmatoriginal Then presentaresultados: Exit Sub
If ordenaasignar = 1 Then
   ReDim elementoaanalizar(numfilasmatoriginal, numcolumnasmatoriginal)
   For i = 1 To numfilasmatoriginal
      For j = 1 To numcolumnasmatoriginal
         elementoaanalizar(i, j).valor = elementooriginal(i, j)
         elementoaanalizar(i, j).serieaquepertenece = i
      Next j
   Next i
End If
If ordenaasignar > 1 Then
   ReDim elementotemporal(UBound(elementoaanalizar, 1) - 1, UBound(elementoaanalizar, 2))
   a = 0
   For i = 1 To UBound(elementoaanalizar, 1)
         'este if se cierra en una línea
         If elementoaanalizar(i, 1).serieaquepertenece = ordenencolumna(1).serieaquepertenece Then GoTo 999
         a = a + 1
         For j = 1 To UBound(elementoaanalizar, 2)
            elementotemporal(a, j).valor = elementoaanalizar(i, j).valor
            elementotemporal(a, j).serieaquepertenece = elementoaanalizar(i, j).serieaquepertenece
         Next j
999  Next i
   'Ya tenemos construída la matriz temporal
   ReDim elementoaanalizar(UBound(elementoaanalizar, 1) - 1, UBound(elementoaanalizar, 2))
   'Hemos destruído el contenido de elemento a analizar Asignamos a elemento a analizar el contenido de elemento temporal
   For i = 1 To UBound(elementotemporal, 1)
      For j = 1 To UBound(elementotemporal, 2)
         elementoaanalizar(i, j).valor = elementotemporal(i, j).valor
         elementoaanalizar(i, j).serieaquepertenece = elementotemporal(i, j).serieaquepertenece
      Next j
   Next i
End If
End Sub

Private Sub Extraercolumna1yordenar()
If UBound(quématriz, 1) > 1 Then
   ReDim ordenencolumna(UBound(quématriz, 1))
Else
   final = True: Establecerordendefinitivo: Exit Sub
End If
For a = 1 To UBound(quématriz, 1)
   For m = 1 To a
      If m = a Then ordenencolumna(m).valor = quématriz(a, 1).valor: _
      ordenencolumna(m).serieaquepertenece = quématriz(a, 1).serieaquepertenece: Exit For
      If quématriz(a, 1).valor > ordenencolumna(m).valor Then Desplazar: Exit For
   Next m
Next a
'Proceso para comprobaciones. Desactivado.
'For i = 1 To UBound(ordenencolumna, 1)
'Print "ordenencolumna"; i; "="; ordenencolumna(i).valor; "con serie"; ordenencolumna(i).serieaquepertenece
'Next i
Empates = 1
For i = 1 To UBound(quématriz, 1)
   If i = 1 Then
      If ordenencolumna(i).valor > ordenencolumna(i + 1).valor Then
         Establecerordendefinitivo
         Exit For                'Se sale sin empates
```

© 2006-2029 aprenderaprogramar.com

```
      End If
    End If
    If i < UBound(quématriz, 1) Then
      If ordenencolumna(i).valor > ordenencolumna(i + 1).valor Then Exit For     ' Se sale hay empates
    End If
    If i < UBound(quématriz, 1) Then Empates = Empates + 1
Next i
If final = False And Empates > 1 Then Dirimirempate
End Sub

Private Sub Desplazar()
t = a + 1
For k = 1 To a - m
    t = t - 1
    ordenencolumna(t).valor = ordenencolumna(t - 1).valor
    ordenencolumna(t).serieaquepertenece = ordenencolumna(t - 1).serieaquepertenece
Next k
ordenencolumna(m).valor = quématriz(a, 1).valor
ordenencolumna(m).serieaquepertenece = quématriz(a, 1).serieaquepertenece
End Sub

Private Sub Dirimirempate()
'Entra en dirimir empate con número de empates definido por la variable Empates
'Caso de que haya iterado, sólo queda una columna y dos o más series de cabeza son absolutamente iguales
If UBound(quématriz, 2) = 1 And Empates > 1 Then Establecerordendefinitivo: Exit Sub
ReDim subelemento(Empates, UBound(quématriz, 2) - 1)
For m = 1 To Empates
    For i = 1 To UBound(quématriz, 1)
      For j = 2 To UBound(quématriz, 2)
        If ordenencolumna(m).serieaquepertenece = quématriz(i, j).serieaquepertenece Then
          subelemento(m, j - 1).valor = quématriz(i, j).valor
          subelemento(m, j - 1).serieaquepertenece = quématriz(i, j).serieaquepertenece
        End If
      Next j
    Next i
Next m
'asigna a quématriz el contenido de subelemento para mandarlo a pasar extraercolumna1yordenar
ReDim quématriz(UBound(subelemento, 1), UBound(subelemento, 2))
For i = 1 To UBound(subelemento, 1)
    For j = 1 To UBound(subelemento, 2)
      quématriz(i, j).valor = subelemento(i, j).valor
      quématriz(i, j).serieaquepertenece = subelemento(i, j).serieaquepertenece
    Next j
Next i
Extraercolumna1yordenar
End Sub

Private Sub Establecerordendefinitivo()
'Entramos en establecer orden definitivo con orden a asignar definido por la variable ordenaasignar
a = 0
For i = 1 To UBound(elementoaanalizar, 1)
    For j = 1 To UBound(elementoaanalizar, 2)
      If elementoaanalizar(i, j).serieaquepertenece = ordenencolumna(1).serieaquepertenece Then
        a = a + 1
        elementomatrizordenada(ordenaasignar, a).valor = elementoaanalizar(i, j).valor
        elementomatrizordenada(ordenaasignar, a).serieaquepertenece = _
        elementoaanalizar(i, j).serieaquepertenece
        'Control desactivado Print "elemento mat ordernada"; ordenaasignar; ","; a; "vale"; _
         elementomatrizordenada(ordenaasignar, a).valor; "y pertenece a la serie"; _
         elementomatrizordenada(ordenaasignar, a).serieaquepertenece
      End If
    Next j
Next i
If final = True Then
    For i = 1 To UBound(quématriz, 2)
      elementomatrizordenada(ordenaasignar, i).valor = quématriz(1, i).valor
      elementomatrizordenada(ordenaasignar, i).serieaquepertenece = quématriz(1, i).serieaquepertenece
    Next i
End If
Definirordenaasignar
Definirmatrizaanalizar
```

© 2006-2029 aprenderaprogramar.com

```
'Asignamos a quématriz el contenido de elementoaanalizar para mandarlo a extraer columna 1 y ordenar
ReDim quématriz(UBound(elementoaanalizar, 1), UBound(elementoaanalizar, 2))
For i = 1 To UBound(elementoaanalizar, 1)
  For j = 1 To UBound(elementoaanalizar, 2)
    quématriz(i, j).valor = elementoaanalizar(i, j).valor
    quématriz(i, j).serieaquepertenece = elementoaanalizar(i, j).serieaquepertenece
  Next j
Next i
If final = True Then Exit Sub
Extraercolumna1yordenar
End Sub

Private Sub presentaresultados()
'Entramos en presentaresultados
Label2 = "La matriz ordenada es esta:" & Chr(13) & Chr(13)
For i = 1 To numfilasmatoriginal
  For j = 1 To numcolumnasmatoriginal
    TPO = TPO & " " & elementomatrizordenada(i, j).valor
  Next j
  Label2 = Label2 & TPO & Chr(13)
  TPO = ""
Next i
End Sub
```

CÓDIGO DEL PROGRAMA ORNUMELP PARA VERSIONES MÁS RECIENTES DE VISUAL BASIC

```
'PROGRAMA ORNUMELP versión 0.1 aprenderaprogramar.com uso exclusivamente didáctico. Prohibido su uso
comercial.
Option Explicit On
Public Class Form1

    Dim a%, m%, t%
    Dim Empates%
    Dim TPO As Object
    Dim final As Boolean
    'Fijamos el tamaño de la matriz declarando estas constantes
    'Admite valores iguales o superiores a 2 (con una sola fila no hay orden que establecer)
    Const numfilasmatoriginal = 4
    Const numcolumnasmatoriginal = 4
    Dim elementooriginal(numfilasmatoriginal, numcolumnasmatoriginal)
    Dim ordenaasignar%

    Private Structure element
        Dim valor As Single
        Dim serieaquepertenece As Integer
    End Structure

    Dim elementoaanalizar(,) As element
    Dim ordenencolumna() As element
    Dim elementomatrizordenada(,) As element
    Dim quématriz(,) As element
    Dim subelemento(,) As element
    Dim elementotemporal(,) As element

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        'Nos permite ordenar la generación de una nueva matriz y su ordenamiento
        'Nos permite ordenar la generación de una nueva matriz y su ordenamiento
        final = False
        ordenaasignar = 0
        Label1.Text = " "
        Me.Form1_Load(Me, New System.EventArgs)
    End Sub
```

© 2006-2029 aprenderaprogramar.com

```
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim i As Integer, j As Integer
        Me.Text = "Ordenar series de números"
        Asignarvaloresamatrizoriginal()
        Definirordenaasignar()
        Definirmatrizaanalizar()
        'Paso a extraer columna 1 y ordenar, indicándole qué matriz usar
        ReDim quématriz(UBound(elementoaanalizar, 1), UBound(elementoaanalizar, 2))
        For i = 1 To UBound(elementoaanalizar, 1)
            For j = 1 To UBound(elementoaanalizar, 2)
                quématriz(i, j).valor = elementoaanalizar(i, j).valor
                quématriz(i, j).serieaquepertenece = elementoaanalizar(i, j).serieaquepertenece
            Next j
        Next i
        Extraercolumna1yordenar()
    End Sub

    Private Sub Asignarvaloresamatrizoriginal()
        Dim i As Integer, j As Integer
        'Asignación de valores directa por el programador, ACTIVADA
        elementooriginal(1, 1) = 7 : elementooriginal(1, 2) = 6 : elementooriginal(1, 3) = 4 :
elementooriginal(1, 4) = 8
        elementooriginal(2, 1) = 6 : elementooriginal(2, 2) = 4 : elementooriginal(2, 3) = 7 :
elementooriginal(2, 4) = 9
        elementooriginal(3, 1) = 6 : elementooriginal(3, 2) = 6 : elementooriginal(3, 3) = 3 :
elementooriginal(3, 4) = 8
        elementooriginal(4, 1) = 4 : elementooriginal(4, 2) = 8 : elementooriginal(4, 3) = 1 :
elementooriginal(4, 4) = 3
        For i = 1 To numfilasmatoriginal
            For j = 1 To numcolumnasmatoriginal
                Label1.Text = Label1.Text & " " & elementooriginal(i, j)
            Next j
            Label1.Text = Label1.Text & Chr(13)
        Next i
        ReDim elementomatrizordenada(numfilasmatoriginal, numcolumnasmatoriginal)
        'Generación de una matriz de valores aleatorios; controlamos el número de empates
        'con el múltiplo del RND; DESACTIVADO
        'Label1.Text = "La matriz a ordenar es esta: " & Chr(13) & Chr(13)
        'For i = 1 To numfilasmatoriginal
        'For j = 1 To numcolumnasmatoriginal
        'Randomize()
        'elementooriginal(i, j) = Int(Rnd() * 9) + 1
        'Label1.Text = Label1.Text & elementooriginal(i, j) & " "
        'Next j
        'Label1.Text = Label1.Text & Chr(13)
        'Next i
    End Sub

    Private Sub Definirordenaasignar()
        ordenaasignar = ordenaasignar + 1
    End Sub

    Private Sub Definirmatrizaanalizar()
        Dim i As Integer, j As Integer
        If ordenaasignar > numfilasmatoriginal Then presentaresultados() : Exit Sub
        If ordenaasignar = 1 Then
            ReDim elementoaanalizar(numfilasmatoriginal, numcolumnasmatoriginal)
            For i = 1 To numfilasmatoriginal
                For j = 1 To numcolumnasmatoriginal
                    elementoaanalizar(i, j).valor = elementooriginal(i, j)
                    elementoaanalizar(i, j).serieaquepertenece = i
                Next j
            Next i
        End If
        If ordenaasignar > 1 Then
            ReDim elementotemporal(UBound(elementoaanalizar, 1) - 1, UBound(elementoaanalizar, 2))
            Me.a = 0
            For i = 1 To UBound(elementoaanalizar, 1)
                'este if se cierra en una línea
                If elementoaanalizar(i, 1).serieaquepertenece = ordenencolumna(1).serieaquepertenece Then
GoTo 999
                Me.a = Me.a + 1
                For j = 1 To UBound(elementoaanalizar, 2)
```

© 2006-2029 aprenderaprogramar.com

```
                    elementotemporal(Me.a, j).valor = elementoaanalizar(i, j).valor
                    elementotemporal(Me.a, j).serieaquepertenece = elementoaanalizar(i,
j).serieaquepertenece
                Next j
999:        Next i
            'Ya tenemos construída la matriz temporal
            ReDim elementoaanalizar(UBound(elementoaanalizar, 1) - 1, UBound(elementoaanalizar, 2))
            'Hemos destruído el contenido de elemento a analizar Asignamos a elemento a analizar el
contenido de elemento temporal
            For i = 1 To UBound(elementotemporal, 1)
                For j = 1 To UBound(elementotemporal, 2)
                    elementoaanalizar(i, j).valor = elementotemporal(i, j).valor
                    elementoaanalizar(i, j).serieaquepertenece = elementotemporal(i,
j).serieaquepertenece
                Next j
            Next i
        End If
    End Sub

    Private Sub Extraercolumna1yordenar()
        Dim i As Integer
        If UBound(quématriz, 1) > 1 Then
            ReDim ordenencolumna(UBound(quématriz, 1))
        Else
            final = True : Establecerordendefinitivo() : Exit Sub
        End If
        For Me.a = 1 To UBound(quématriz, 1)
            For Me.m = 1 To Me.a
                If Me.m = Me.a Then ordenencolumna(Me.m).valor = quématriz(Me.a, 1).valor _
                  : ordenencolumna(Me.m).serieaquepertenece = quématriz(Me.a, 1).serieaquepertenece :
Exit For
                If quématriz(Me.a, 1).valor > ordenencolumna(Me.m).valor Then Desplazar() : Exit For
            Next Me.m
        Next Me.a
        'Proceso para comprobaciones. Desactivado.
        'For i = 1 To UBound(ordenencolumna, 1)
        'Print "ordenencolumna"; i; "="; ordenencolumna(i).valor; "con serie";
ordenencolumna(i).serieaquepertenece
        'Next i
        Empates = 1
        For i = 1 To UBound(quématriz, 1)
            If i = 1 Then
                If ordenencolumna(i).valor > ordenencolumna(i + 1).valor Then
                    Establecerordendefinitivo()
                    Exit For                    'Se sale sin empates
                End If
            End If
            If i < UBound(quématriz, 1) Then
                If ordenencolumna(i).valor > ordenencolumna(i + 1).valor Then Exit For ' Se sale hay
empates
            End If
            If i < UBound(quématriz, 1) Then Empates = Empates + 1
        Next i
        If final = False And Empates > 1 Then Dirimirempate()
    End Sub

    Private Sub Desplazar()
        Dim k As Integer
        Me.t = Me.a + 1
        For k = 1 To Me.a - Me.m
            Me.t = Me.t - 1
            ordenencolumna(Me.t).valor = ordenencolumna(Me.t - 1).valor
            ordenencolumna(Me.t).serieaquepertenece = ordenencolumna(Me.t - 1).serieaquepertenece
        Next k
        ordenencolumna(Me.m).valor = quématriz(Me.a, 1).valor
        ordenencolumna(Me.m).serieaquepertenece = quématriz(Me.a, 1).serieaquepertenece
    End Sub

    Private Sub Dirimirempate()
        Dim i As Integer, j As Integer
        'Entra en dirimir empate con número de empates definido por la variable Empates
        'Caso de que haya iterado, sólo queda una columna y dos o más series de cabeza son absolutamente
iguales
        If UBound(quématriz, 2) = 1 And Empates > 1 Then Establecerordendefinitivo() : Exit Sub
```

© 2006-2029 aprenderaprogramar.com

```
        ReDim subelemento(Empates, UBound(quématriz, 2) - 1)
        For Me.m = 1 To Empates
            For i = 1 To UBound(quématriz, 1)
                For j = 2 To UBound(quématriz, 2)
                    If ordenencolumna(Me.m).serieaquepertenece = quématriz(i, j).serieaquepertenece Then
                        subelemento(Me.m, j - 1).valor = quématriz(i, j).valor
                        subelemento(Me.m, j - 1).serieaquepertenece = quématriz(i, j).serieaquepertenece
                    End If
                Next j
            Next i
        Next Me.m
        'asigna a quématriz el contenido de subelemento para mandarlo a pasar extraercolumna1yordenar
        ReDim quématriz(UBound(subelemento, 1), UBound(subelemento, 2))
        For i = 1 To UBound(subelemento, 1)
            For j = 1 To UBound(subelemento, 2)
                quématriz(i, j).valor = subelemento(i, j).valor
                quématriz(i, j).serieaquepertenece = subelemento(i, j).serieaquepertenece
            Next j
        Next i
        Extraercolumna1yordenar()
    End Sub

    Private Sub Establecerordendefinitivo()
        Dim i As Integer, j As Integer
        'Entramos en establecer orden definitivo con orden a asignar definido por la variable
ordenaasignar
        Me.a = 0
        For i = 1 To UBound(elementoaanalizar, 1)
            For j = 1 To UBound(elementoaanalizar, 2)
                If elementoaanalizar(i, j).serieaquepertenece = ordenencolumna(1).serieaquepertenece Then
                    Me.a = Me.a + 1
                    elementomatrizordenada(ordenaasignar, Me.a).valor = elementoaanalizar(i, j).valor
                    elementomatrizordenada(ordenaasignar, Me.a).serieaquepertenece = _
                    elementoaanalizar(i, j).serieaquepertenece

                End If
            Next j
        Next i
        If final = True Then
            For i = 1 To UBound(quématriz, 2)
                elementomatrizordenada(ordenaasignar, i).valor = quématriz(1, i).valor
                elementomatrizordenada(ordenaasignar, i).serieaquepertenece = quématriz(1,
i).serieaquepertenece
            Next i
        End If
        Definirordenaasignar()
        Definirmatrizaanalizar()
     'Asignamos a quématriz el contenido de elementoaanalizar para mandarlo a extraer columna 1 y ordenar
        ReDim quématriz(UBound(elementoaanalizar, 1), UBound(elementoaanalizar, 2))
        For i = 1 To UBound(elementoaanalizar, 1)
            For j = 1 To UBound(elementoaanalizar, 2)
                quématriz(i, j).valor = elementoaanalizar(i, j).valor
                quématriz(i, j).serieaquepertenece = elementoaanalizar(i, j).serieaquepertenece
            Next j
        Next i
        If final = True Then Exit Sub
        Extraercolumna1yordenar()
    End Sub

    Private Sub presentaresultados()
        Dim i As Integer
        Dim j As Integer
        'Entramos en presentaresultados
        Label2.Text = "La matriz ordenada es esta:" & Chr(13) & Chr(13)
        For i = 1 To numfilasmatoriginal
            For j = 1 To numcolumnasmatoriginal
                TPO = TPO & " " & elementomatrizordenada(i, j).valor
            Next j
            Label2.Text = Label2.Text & TPO & Chr(13)
            TPO = ""
        Next i
    End Sub

End Class
```

© 2006-2029 aprenderaprogramar.com

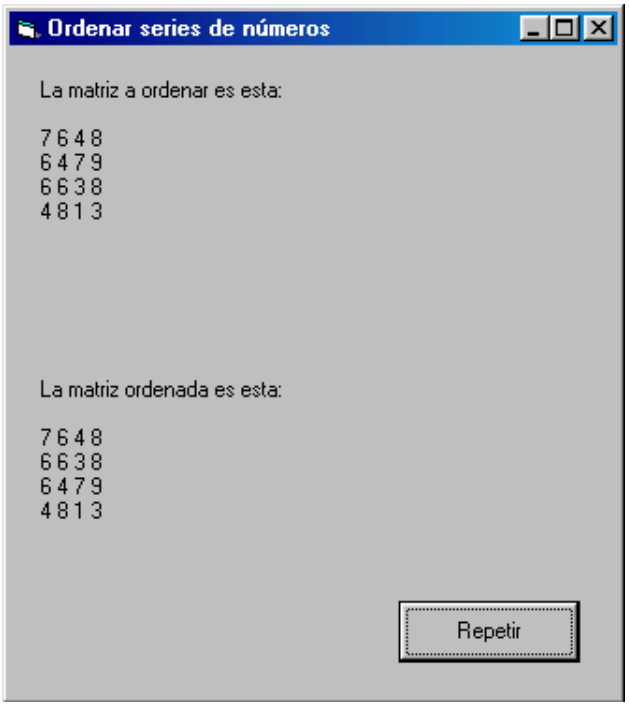
Comentarios: El programador controla la dimensión de la matriz a ordenar a través de la declaración Const numfilasmatoriginal = 4 y Const numcolumnasmatoriginal = 4.

Por otro lado, se controla el intervalo de valores para la generación de datos aleatorios a través de la expresión.

elementooriginal(i, j) = Int(Rnd * 9) + 1

Esta expresión constituye una forma de regular el número de empates que queremos que aparezcan. Si el número que multiplica a Rnd es bajo habrá un mayor número de repeticiones en los elementos de la matriz.

El resultado del programa será de este tipo:



Próxima entrega: CU00359A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com

FINAL DEL CURSO "PROGRAMACIÓN EN VISUAL BASIC NIVEL I"

Con esta entrega llegamos al final del curso "Programación en Visual Basic nivel I". Esperamos que haya sido un curso útil y ameno para todas las personas que lo hayan seguido. Y como en todo final, cabe hacer algunas consideraciones especiales.

- Gracias al equipo humano de aprenderaprogramar.com que ha hecho posible su publicación, y en especial a César Hernández, Javier Roa, Jorge Maestro, Manuel Tello, Walter Sagástegui y Enrique González.

- Gracias a todas las personas que de una u otra forma han participado enviando propuestas de mejora, comentarios, avisos de erratas, etc. y a los alumnos que han seguido el curso en las modalidades de tickets on-line y tutorizado on-line.

- A todos los que no han participado pero han seguido el curso de forma gratuita a través de la web, desde aprenderaprogramar.com les agradeceríamos nos hicieran llegar una opinión o propuesta de mejora sobre el mismo, bien a través de correo electrónico a contacto@aprenderaprogramar.com , bien a través de los foros. Todas las opiniones son bienvenidas y nos sirven para mejorar.

- A quienes hayan seguido el curso de forma gratuita y piensen que los contenidos son de calidad y que merece dar un pequeño apoyo económico para que se puedan seguir ofreciendo más y mejores contenidos en este sitio web, les estaremos muy agradecidos si realizan una pequeña aportación económica en forma de donación pulsando sobre el enlace que aparece en la página principal de aprenderaprogramar.com.

Quienes tengan interés en proseguir formándose en el área de programación con aprenderaprogramar.com, pueden hacerlo inscribiéndose en el curso tutorizado on-line "Aprender programación Java desde cero", en caso de que deseen contar con un tutor que les orientará, guiará y resolverá dudas. Pueden ampliar información en esta URL: http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=64&Itemid=87

Quienes quieran proseguir con el curso "Aprender programación Java desde cero" por su cuenta, pueden acceder a las entregas en esta URL:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188

A todos los que nos han leído y nos siguen, gracias. ¡Nos vemos en el próximo!

El equipo de aprenderaprogramar.com

© 2006-2029 aprenderaprogramar.com